Mathematical topics in coding and cryptology

Fletcher, Matthew

Award date:
1992

Awarding institution:
University of Bath

Mathematical Topics in Coding and Cryptology

submitted by Matthew Fletcher [1]

for the degree of PhD

of the University of Bath

1992

M. Fletcher

[1] work supported by the SERC

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Several properties of functions have been deemed desirable for encryption. In this work we focus particularly on three such properties: balance, uncorrelatedness and the strict avalanche criterion, proving several new results.

We also introduce the idea of computationally equivalent functions, and develop a fast algorithm for determining whether two functions are equivalent. We make use of the program CAYLEY used for computational group theory. The S-boxes of the Data Encryption Standard are then analysed from the viewpoint of computationally equivalent and uncorrelated functions.

Finally we examine finite analogues of the Mandelbrot Set by considering iterated functions over finite rings and fields, rather than the complex plane. We define perfectly cyclic numbers and prove results concerning the cyclicity of certain primes.

Contents

Introduction

Any function used for encryption must be hard to invert. In other words it must be as difficult as possible to determine any information about the input when the output is known, without some secret knowledge such as a password. The study of such encryptions has identified many desirable properties which functions should possess in order to withstand a variety of attacks. There is inevitably some conflict between all the properties we would like a function to have and some may have to be sacrificed so that other criteria may be satisfied.

The first two chapters are mainly concerned with three of the most important properties: Balance, the Strict Avalanche Criterion and Uncorrelatedness. Using combinatorial and probabilistic arguments, we derive many new results concerning such functions, in particular an estimation of the number of functions satisfying the SAC, and the sizes of various families of uncorrelated functions.

In Chapter 3 we introduce the notion of computationally equivalent functions. These are functions which are essentially the same, one being derived from another by a permutation and complementation of the inputs. By constructing a group theoretic definition of equivalence we develop an algorithm for testing the

equivalence of functions and use it to search for relationships between the S-box functions used in the DES. The S-boxes are also analysed for uncorrelatedness and we draw attention to an anomaly in one of the permutations appearing in the cipher.

Chapter 4 is concerned with iterated functions taken modulo an integer. In trying to establish a finite analogue of the Mandelbrot set, we investigate a new concept of perfect cyclicity. We prove several results determining which primes are perfectly cyclic by considering the distribution of quadratic residues over finite fields. Algebraic number theory and the theory of elliptic curves over finite fields are the main mathematical areas of investigation.

Chapter 1

Balanced Functions and the Strict Avalanche Criterion

1.1 Introduction

A balanced function is one in which all outputs occur equally often. Balance is the most natural of all the criteria a function must satisfy to maximise the difficulty of computing its inverse. Clearly any departure from balance can only help a statistical attack.

The condition for a function to satisfy the strict avalanche criterion (SAC) was first stated by Webster and Tavares in [22]:

> ...each of its output bits should change with a probability one half whenever a single input bit ...is complemented.

The strictness refers to the insistence that the probability of each output changing is precisely one half. From a mathematical viewpoint this definition is convenient but it is clear that a genuine encryption function would only be required to approximately satisfy the SAC. Furthermore, it is conceivable that only using functions satisfying the SAC would so reduce the functions available that a brute force attempt to break a cryptosystem would be successful. In Section 1.6 we obtain a lower bound for the number of Boolean functions satisfying the SAC and they are increasingly rare as the number of input bits increases.

The reason the SAC is a desirable property is to prevent key-clustering attacks. If a one-way function were locally linear it would be possible to find an $\underline{x}$ such that $f(\underline{x}) = \underline{y}$ for a given $\underline{y}$ as follows. First a number of pairs of the form $(\underline{z}, f(\underline{z}))$ are computed, then for each pair with $f(\underline{z})$ close to $\underline{y}$ a large number of inputs close to $\underline{z}$ are tested, hoping to find the pair $(\underline{x}, \underline{y})$. By a vector being close to another we simply mean differing in a small number of places. Satisfaction of the SAC ensures that such an attack is no more likely to succeed than a completely random sample.

In this chapter the SAC is characterised in terms of a function's partial derivatives. This allows many results concerning functions satisfying the SAC to be proved more easily and we include a new estimation of the number of Boolean functions on n bits satisfying the SAC.

1.2 Definitions

Let $\mathbf{Z}_2$ denote the field GF(2) and $\mathbf{Z}_2^n$ the n-dimensional vector space over $\mathbf{Z}_2$.

Every function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ can be uniquely expressed as a sum of monomials in n variables, $x_1, \ldots, x_n$, where each monomial is a product of between 0 and n of the x_i s. This is known as its algebraic normal form. Let the weight of such a monomial be the number of x_i s appearing in the product. We shall refer to this representation as the polynomial of a function. Implicit in this definition is the assumption that multiple occurrences of the same monomial are prohibited. A function and its polynomial may be used interchangeably provided the context does not allow ambiguities.

Let $f_i : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ be defined by $f_i(\underline{x}) = x_i$, then f_i is the i–th bit selector function. It is also useful to define the unit function $\mathbf{1} : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ as $\mathbf{1}(\underline{x}) = 1$ for all $\underline{x} \in \mathbf{Z}_2^n$.

Let $\mathbf{V}_n$ be the $(n+1)$-dimensional vector space generated by $< f_1, \ldots, f_n, \mathbf{1} >$ over the field $\mathbf{Z}_2$. It is clear that $\mathbf{V}_n$ consists of all functions from $\mathbf{Z}_2^n$ to $\mathbf{Z}_2$ which have polynomials of the form $\lambda_0 + \sum_{i=1}^{n} \lambda_i x_i$ with $\lambda_i \in \mathbf{Z}_2$.

1.2.1 Balance

A function is balanced if each of its outputs occurs equally often, as the input ranges over the function's domain. For a function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$, f is balanced if and only if

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) = 2^{n-1}, \tag{1.1}$$

since precisely half of the 2^n inputs will yield one as output.

This definition is not as clear cut as it may first appear. We are summing values from the field $\mathbf{Z}_2$ but evaluating the result in the ring of integers $\mathbf{Z}$. This notation is used widely throughout the thesis and in order to reduce ambiguity the symbol $\oplus$ has been used where addition in $\mathbf{Z}_2$ is required.

1.2.2 The Strict Avalanche Criterion

The SAC can be expressed more formally as

$f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2^k$ satisfies the SAC if and only if

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) \oplus f(\underline{x} \oplus \underline{c}_i) = (2^{n-1}, \ldots, 2^{n-1}) \quad \forall i : 1 \leq i \leq n. \qquad (1.2)$$

We use $\underline{c}_i$ to denote the n-bit vector with a 1 in the i-th position and zeros elsewhere.

A necessary and sufficient condition for a function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2^k$ to satisfy the SAC is that each output bit independently satisfies the SAC, so without loss of generality we can restrict our attention to the case $k = 1$. All evaluations of functions will therefore be done in the field $\mathbf{Z}_2$. A function which only takes values in $\mathbf{Z}_2$ is referred to as Boolean.

Forré [7] introduced the SAC of order m. A function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ is said to satisfy the SAC of order $m : 0 \leq m \leq (n-2)$ if the subfunction obtained by keeping any 0 to m of the input bits constant also satisfies the SAC.

For $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ and a given $i : 1 \leq i \leq n$, we can write

$$f(\underline{x}) = x_i g_i(\underline{x}') \oplus h_i(\underline{x}'), \tag{1.3}$$

where $g_i, h_i : \mathbf{Z}_2^{n-1} \rightarrow \mathbf{Z}_2$, and $\underline{x}'$ is the vector consisting of $x_1, \ldots, x_n$ except for x_i.

Equation (1.3) separates $f(\underline{x})$ into all monomials including x_i, and all monomials without x_i. Now we can see that

$$f(\underline{x} \oplus \underline{c}_i) = (x_i \oplus 1) g_i(\underline{x}') \oplus h_i(\underline{x}'). \tag{1.4}$$

Now using (1.3) and (1.4) we can rewrite the left-hand-side of condition (1.2) as

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) \oplus f(\underline{x} \oplus \underline{c}_i) = \sum_{\underline{x} \in \mathbf{Z}_2^n} g_i(\underline{x}') = \sum_{\underline{x} \in \mathbf{Z}_2^n} \frac{\partial f}{\partial x_i}, \tag{1.5}$$

where the partial differentiation is defined formally on the polynomial representation of $f(\underline{x})$ in the natural way. Therefore,

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) \oplus f(\underline{x} \oplus \underline{c}_i) = 2^{n-1} \text{ if and only if } \sum_{\underline{x} \in \mathbf{Z}_2^n} \frac{\partial f}{\partial x_i} = 2^{n-1}. \tag{1.6}$$

Thus, a function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfies the SAC if and only if all its partial derivatives are balanced. This observation was noted in [17] and independently by the author. We now use this observation to prove several results about functions satisfying the SAC.

1.3 Preliminary Results

Theorem 1.1

If $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfies the SAC and $g \in \mathbf{V}_n$, then $f \oplus g$ satisfies the SAC.

Proof

By definition,

$$g(\underline{x}) = \lambda_0 \oplus \sum_{j=1}^{n} \lambda_j x_j \text{ for } \lambda_j \in \mathbf{Z}_2.$$

Therefore,

$$\frac{\partial}{\partial x_i}(f \oplus g) = \frac{\partial f}{\partial x_i} \oplus \lambda_i,$$

which is balanced, since adding 0 or 1 to a balanced function preserves its balance.

Corollary

If $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfies the SAC of order m and $g \in \mathbf{V}_n$, then $f \oplus g$ satisfies the SAC of order m.

Proof

As above, noting that any subfunction obtained from an element of $\mathbf{V}_n$ is also an element of $\mathbf{V}_n$.

Theorem 1.2

If $f \in \mathbf{V}_n$ and f is neither of the constant functions **1** or **01**, then f is balanced.

Proof

Given such an f we can write

$$f(\underline{x}) = \lambda \oplus \sum_{i \in \mathrm{I}} x_i \text{ for some } \emptyset \neq \mathrm{I} \subseteq \{1, 2, \ldots, n\}, \lambda \in \mathbf{Z}_2,$$

and the proof is by induction on the size of I which we denote by #I.

Base Case: $\#\mathrm{I} = 1$

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) = \sum_{\underline{x} \in \mathbf{Z}_2^n} (\lambda \oplus x_i) = 2^{n-1}$$

Inductive Step:

Assume that all functions in $\mathbf{V}_n$ for which $\#\mathrm{I} = N$ are indeed balanced or equal to **0** or **1**. Now suppose that

$$f(\underline{x}) = \lambda \oplus \sum_{i \in \mathrm{I}} x_i, \qquad \text{with } \#\mathrm{I} = N + 1.$$

Rewriting,

$$f(\underline{x}) = \lambda \oplus \sum_{i \in \mathrm{I}'} x_i \oplus x_j \text{ for some } j \in \mathrm{I} \text{ where } \mathrm{I}' = \mathrm{I} - \{j\}.$$

Therefore,

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) = \sum_{\underline{x} \in \mathbf{Z}_2^n} (\lambda \oplus \sum_{i \in \mathrm{I}'} x_i \oplus x_j) = \sum_{\underline{x} \in \mathbf{Z}_2^n} \lambda \oplus (\sum_{i \in \mathrm{I}'} x_i \oplus x_j)$$

$$= \sum_{\substack{\underline{x} \in \mathbf{Z}_2^n \\ x_j = 0}} \lambda \oplus (\sum_{i \in \mathrm{I}'} x_i \oplus x_j) \quad + \quad \sum_{\substack{\underline{x} \in \mathbf{Z}_2^n \\ x_j = 1}} \lambda \oplus (\sum_{i \in \mathrm{I}'} x_i \oplus x_j)$$

$$= \frac{1}{2} \sum_{\underline{x} \in \mathbf{Z}_2^n} \sum_{i \in \mathrm{I}'} x_i \quad + \quad \frac{1}{2} \sum_{\underline{x} \in \mathbf{Z}_2^n} (\sum_{i \in \mathrm{I}'} x_i \oplus 1) = 2^{n-2} + 2^{n-2} = 2^{n-1}.$$

Hence by induction, theorem holds for all I such that $\#\mathrm{I} \geq 1$.

Theorem 1.3

A balanced function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ cannot have the term $x_1 x_2 \ldots x_n$ in its polynomial for $n > 1$.

Proof

Consider the following method of constructing the polynomial of a function:

For each $\underline{x}_k$ such that $f(\underline{x}_k) = 1$ form the product

$$p_k = \prod_{i=1}^{n} (x_i \oplus a_i) \quad \text{where} \begin{cases} a_i = 1, & \text{if } x_i = 0 \text{ in } \underline{x}_k \\ a_i = 0, & \text{if } x_i = 1 \text{ in } \underline{x}_k \end{cases},$$

then the polynomial of f is $\sum p_k$.

If each p_k is expanded, it will have the term $x_1 x_2 \ldots x_n$. For a balanced function there will be 2^{n-1} such p_ks. All terms appearing an even number of times in $\sum p_k$ will vanish, because the polynomial is evaluated in $\mathbf{Z}_2$, so $x_1 x_2 \ldots x_n$ will vanish provided that 2^{n-1} is even, in other words provided that $n > 1$.

Corollary

A function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfying the SAC cannot have the term $x_1 x_2 \ldots x_n$ in its polynomial for $n > 2$.

Proof

If f has a term of weight n, $n > 2$, then all $\dfrac{\partial f}{\partial x_i}$s will have the corresponding term of weight $(n-1)$ with $(n-1) > 1$, and so could not be balanced.

Theorem 1.4

A function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfying the SAC of order m cannot have any terms of weight greater than $(n-m-1)$ in its polynomial for $m < n-2$.

Proof

Let the polynomial of $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ have a term of weight k where $k > (n-m-1)$. Without loss of generality, suppose that the term is $x_1 \ldots x_k$. Set $x_{k+1}, \ldots, x_n$ equal to 0, and set $x_{n-m+1}, \ldots, x_k$ equal to 1. So in total, the values of $(n-k)+k-(n-m) = m$ input bits have been fixed.

Now consider the subfunction $\hat{f} : \mathbf{Z}_2^{n-m} \rightarrow \mathbf{Z}_2$ thus obtained. It has the term $x_1 x_2 \ldots x_{n-m}$ and $(n-m) > 2$, so, by the Corollary to Theorem 1.3, it cannot satisfy the SAC.

Theorem 1.4 has been stated and proved by many researchers independently, including the author. The earliest occurrence appears to be in [10] where it is attributed to Schroeppel.

When a function satisfies the SAC of order m, then it automatically satisfies the SAC of all lower orders. This means that although Theorem 1.4 does not apply for the case $m = (n-2)$, a function f satisfying the SAC of order $(n-2)$ will of course satisfy SAC of order $(n-3)$. We can therefore deduce that there are no terms of weight greater than 2 in its polynomial.

Theorem 1.5

A function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ whose polynomial consists only of terms of weight 2 or less satisfies the SAC if and only if each x_i appears at least once in a term of weight 2.

Proof

Referring to terms of weight 2 as quadratics, if an x_i did not appear in any quadratic then $\frac{\partial f}{\partial x_i}$ would be constant, and therefore not balanced, so the condition is necessary. If each x_i appears in a quadratic, then

$$\frac{\partial f}{\partial x_i} = \lambda \oplus \sum_{j \in J} x_j \quad \text{for some } J : \#J \geq 1, \text{ and some } \lambda \in \mathbf{Z}_2$$

and so, by Theorem 1.2, $\frac{\partial f}{\partial x_i}$ is balanced, therefore the condition is sufficient.

Corollary

A function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ whose polynomial consists only of terms of weight 2 or less satisfies the SAC of order m if and only if each x_i appears in at least $(m+1)$ quadratics.

Proof

If an x_i only appeared in m or fewer quadratics, then by fixing m selected input bits, the subfunction obtained would not contain any quadratics in which x_i appeared, and so by Theorem 1.5 could not satisfy the SAC.

If each x_i appears in at least $(m+1)$ quadratics, then any subfunction obtained by fixing any m input bits will still have every remaining x_i appearing in at least one quadratic.

So the condition is necessary and sufficient.

This result immediately gives a class of functions satisfying the SAC of any desired order. An important observation is that terms of large weight reduce

the order of the SAC which can be satisfied.

Theorem 1.6

If a function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfies the SAC, then f takes each of the values 0 and 1 on at least a quarter, and no more than three-quarters, of its domain.

Proof

Define $X_0 = \{\underline{x} \in \mathbf{Z}_2^n : f(\underline{x}) = 0\}$ and $X_1 = \{\underline{x} \in \mathbf{Z}_2^n : f(\underline{x}) = 1\}$. Consider all unordered pairs of vectors $\{\underline{v}, \underline{w}\}$ with $\underline{v}, \underline{w} \in \mathbf{Z}_2^n$ and $\underline{v}$ and $\underline{w}$ differing in precisely one place. There are $n.2^{n-1}$ such pairs, and, since f satisfies the SAC, exactly half of them have the property that the vectors are mapped to different values. Therefore, at least $n.2^{n-2}$ pairs must contain an element from both X_0 and X_1. Each element of X_0 and X_1 appears in n pairs, therefore $\#X_0, \#X_1 \geq 2^{n-2}$. Since $\#X_0 + \#X_1 = 2^n$, we immediately obtain the result

$$\frac{1}{4}2^n \leq \#X_0, \#X_1 \leq \frac{3}{4}2^n. \tag{1.7}$$

1.4 Characterisation of functions satisfying the SAC of order $(n-2)$

From Theorem 1.4, we know that a function satisfying the SAC of order $(n-2)$ cannot have any terms of weight greater than 2, and from the Corollary to Theorem 1.5, each x_i must appear in at least $(n-1)$ quadratics, in other words, all $\binom{n}{2}$ quadratics must be present.

Therefore the functions $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ which satisfy the SAC of order $(n-2)$ are those of the form $g(\underline{x}) \oplus \mathbf{V}_n$, where

$$g(\underline{x}) = \sum_{1 \leq i < j \leq n} x_i x_j, \tag{1.8}$$

and there are $\#\mathbf{V}_n = 2^{n+1}$ of them.

For example, the 16 functions from $\mathbf{Z}_2^3 \rightarrow \mathbf{Z}_2$ satisfying the SAC of order 1 are:

$x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \oplus \lambda_1 x_1 \oplus \lambda_2 x_2 \oplus \lambda_3 x_3 \oplus \lambda_0$ where $\lambda_i \in \mathbf{Z}_2$, for $0 \leq i \leq 3$.

The number of functions satisfying the SAC of order $(n-2)$ was first proved by Lloyd in [13] using a different characterisation.

1.5 Characterisation of functions satisfying the SAC of order $(n-3)$

If $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfies the SAC of order $(n-3)$ then, by Theorem 1.4, it has no terms of weight greater than 2 in its polynomial, and by the Corollary to Theorem 1.5 each x_i must appear in at least $(n-2)$ quadratics. This condition is clearly sufficient, since after fixing up to $(n-3)$ input bits we must still have each remainging x_i appearing in at least one quadratic.

Let T_n be the number of functions of the form $\sum_{1 \leq i < j \leq n} x_i x_j$, where each x_i appears in at least $(n-2)$ quadratics. To such a function we can add any member of $\mathbf{V}_n$ without affecting its avalanche properties, so the number of functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfying the SAC of order $(n-3)$ is $2^{n+1} T_n$.

Now we can determine T_n.

Consider all the quadratics in which x_n appears; it either appears in $(n-1)$ or $(n-2)$ of them.

If x_n appears in $(n-1)$ quadratics then it is paired with every other x_i once, and so the remaining $(n-1)$ x_is must appear at least another $(n-3)$ times. This can happen in T_{n-1} ways.

If x_n appears in $(n-2)$ quadratics then there exists some x_k which does not occur paired with x_n. Now x_k and x_n must be paired with all other x_is, and the remaining $(n-2)$ x_is must appear at least another $(n-4)$ times. This can happen in T_{n-2} ways, and there are $(n-1)$ ways of choosing which x_k is not paired with x_n.

Thus, we obtain the recurrence relation

$$T_n = T_{n-1} + (n-1)T_{n-2} \qquad \text{for } n \geq 5. \tag{1.9}$$

By direct calculation, $T_3 = 4$ and $T_4 = 10$. Define $T_0 = 1, T_1 = 1, T_2 = 2$ so the T_n is consistently defined for all $n \geq 0$.

A closed form for T_n has proved difficult to find, but we can calculate its exponential generating function, defined by

$$G(z) = \sum_{n \geq 0} T_n \frac{z^n}{n!}. \tag{1.10}$$

Re-writing using (1.9), we have

$$\begin{aligned} G(z) &= \sum_{n \geq 0} T_{n+2} \frac{z^n}{(n+1)!} - \sum_{n \geq 0} T_{n+1} \frac{z^n}{(n+1)!} \\ &= \frac{G'(z) - T_1}{z} - \frac{G(z) - T_0}{z}. \end{aligned} \tag{1.11}$$

Therefore,

$$\frac{G'(z)}{G(z)} = z + 1. \tag{1.12}$$

Solving (1.12) using the initial conditions , we deduce that

$$G(z) = e^{\frac{1}{2}z^2+z}. \tag{1.13}$$

Some small values of T_n and $2^{n+1}T_n$ are given in Table 1.1.

n	3	4	5	6	7	8	9	10
T_n	4	10	26	76	232	764	2620	9496
$2^{n+1}T_n$	64	320	1664	9728	59392	391168	2682880	19447808

Table 1.1: Small Values of T_n and $2^{n+1}T_n$

The number of functions satisfying the SAC of order $(n-3)$ was first proved by Lloyd in [14] using a different characterisation.

1.6 Estimation of number of functions satisfying the SAC

Let us define

$$F(n) = 2^{2^n}2^{n-1}\left[2^{-2^{n-1}}\binom{2^{n-1}}{2^{n-2}}\right]^n, \tag{1.14}$$

then numerical evidence suggests that $F(n)$ is a good approximation to the expected number of functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfying the SAC.

Conjecture 1.1

Let $SAC(n)$ denote the number of functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ which satisfy the SAC, then

$$F(n) \rightarrow SAC(n) \qquad \text{as } n \rightarrow \infty.$$

We present an informal argument below in support of this conjecture, showing how the definition of $F(n)$ arose.

As observed, a function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfies the SAC if and only if $\frac{\partial f}{\partial x_i}$ is balanced $\forall i : 1 \leq i \leq n$. We may, therefore, try to find such a function by constructing a set of consistent partial derivatives. By consistent, it is understood that

$$\frac{\partial}{\partial i}(\frac{\partial f}{\partial x_j}) = \frac{\partial}{\partial j}(\frac{\partial f}{\partial x_i}) \quad \forall i, j : 1 \leq i, j \leq n. \tag{1.15}$$

Only a consistent set of derivatives could have been derived from a function.

Initially, given a function f, consider one of its partial derivatives $\frac{\partial f}{\partial x_1}$. The probability of this being balanced is $2^{-2^{n-1}}\binom{2^{n-1}}{2^{n-2}}$, the fraction of Boolean functions on $(n-1)$ input bits which are balanced. Now, assuming that we were successful, consider the probability that $\frac{\partial f}{\partial x_2}$, is also balanced. As $\frac{\partial f}{\partial x_1}$ is balanced, it does not contain the term $x_2 x_3 \ldots x_n$, and so none of the other partial derivatives could contain the corresponding terms of weight $(n-1)$. This doubles the chance of finding a balanced function, since we are immediately reducing our search space by half. Therefore, the probability of $\frac{\partial f}{\partial x_2}$, and all other partial derivatives, being balanced is $2 \times 2^{-2^{n-1}}\binom{2^{n-1}}{2^{n-2}}$.

If we now make the simplifying, but not correct, assumption that a function is balanced or unbalanced with respect to each input bit independently, then we deduce that the probability of choosing a set of consistent partial derivatives, all balanced, is

$$2^{n-1}\left[2^{-2^{n-1}}\binom{2^{n-1}}{2^{n-2}}\right]^n.$$

The accuracy of $F(n)$ as an approximation to $SAC(n)$ depends entirely on how

large an error is introduced by this step. This error has proved difficult to quantify which explains why Conjecture 1.1 is not a theorem. We can say that $F(n)$ will underestimate, as a function with $\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_{n-1}}$ balanced is more likely to have $\frac{\partial f}{\partial x_n}$ also balanced, than a function where $\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_{n-1}}$ are strongly unbalanced, as it will be easier to find new partial derivatives which are both balanced, and consistent with existing partial derivatives. However, as n increases, we conjecture that this correlation between distinct partial derivatives will decrease, as each component has a lessening influence on the others.

When $n > 4$, the number of functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ is too large to allow exhaustive testing within a reasonable time span. By sampling a random selection, however, we can estimate the number of functions satisfying the SAC, and compare these estimates with Conjecture 1.1's predicted values.

When $n = 5$, Conjecture 1.1 predicts that the proportion satisfying the SAC will be

$$2^4 \left[2^{-2^4} \binom{2^4}{2^3}\right]^5,$$

or about 1 in 214. A sample of 20,000 revealed 127 functions satisfying the SAC, about 1 in 157.

Similarly, for the case $n = 6$, Conjecture 1.1 predicts a proportion of

$$2^5 \left[2^{-2^5} \binom{2^5}{2^4}\right]^6,$$

or about 1 in 4159. A sample of 100,000 found 29, about 1 in 3448.

The case $n = 7$ is the last that could be feasibly computed. The predicted proportion of functions satisfying the SAC is only

$$2^6 \left[2^{-2^6} \binom{2^6}{2^5}\right]^7,$$

about 1 in 164,000. A sample of 1,000,000 functions was taken and 9 were found satisfying the SAC, but clearly a larger sample would be required for a reasonably accurate estimate. This sample took about several days however, and larger samples were ruled out as consuming too much computer time.

1.7 Error Estimation in the Sampling

We would like to estimate the error introduced by sampling and give a confidence interval for the estimated number of functions satisfying the SAC, rather than an imprecise figure. If we take a sample of size T from a population with a proportion p satisfying the SAC, then the expected number of functions satisfying the SAC in the sample, E say, will clearly have a binomial distribution, with

$$\mathcal{P}[E = k] = \binom{T}{k} p^k (1-p)^{T-k}.$$

As our sample is size is large we can obtain an excellent approximation to this binomial distribution with the normal distribution $N(pT, p(1-p)T)$, where pT is the mean, and $p(1-p)T$ the variance. Let our observed proportion be $\hat{p}$, then to find a confidence interval of 95% for p, we must calculate the upper and lower proportions p_u and p_l for which our observed value $\hat{p}$ would be so small with a probability of 2.5% or less, and so large with a probability of 2.5% or less respectively. From the definitions of p_u and p_l we can immediately deduce the following equations:

$$\hat{p}T = p_u T - 1.96\sqrt{p_u(1-p_u)T}, \qquad \hat{p}T = p_l T + 1.96\sqrt{p_l(1-p_u)T}.$$

We conclude that the true proportion p lies in the interval (p_l, p_u) with probability 95%. Our estimate for the number of functions in $\mathbf{Z}_2^n \to \mathbf{Z}_2$ satisfying the

SAC will therefore be the interval $(\frac{2^{2^n}p_l}{T}, \frac{2^{2^n}p_u}{T})$.

All the above results and together with calculations of confidence intervals are summarised in Table 1.2. The results generally support Conjecture 1.1, although not too much reliance should be placed in the estimate for $n = 7$.

n	$F(n)$	$SAC(n)$	$F(n)/SAC(n)$
2	8	8	1.00
3	54	64	0.84
4	2931	4128	0.71
5	2.01×10^7	$(2.73 \pm 0.47) \times 10^7$	≈ 0.7
6	4.44×10^{15}	$(5.35 \pm 1.98) \times 10^{15}$	≈ 0.8
7	2.08×10^{33}	$(3.1 \pm 2.1) \times 10^{33}$	≈ 0.7

Table 1.2: The number of functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ satisfying the SAC

Chapter 2

Uncorrelated Functions

2.1 Introduction

A standard approach when attempting to cryptanalyse an encryption function is to examine the behaviour when one or more of the input bits is held constant. Clearly the subfunctions obtained must mimic their parent function's complexity as closely as possible. If we insist that it is not possible to infer any information about any of input bits for a given output, we are in fact requiring that the subfunctions obtained by setting an input bit to 0 or 1 are equally unbalanced.

Using this idea we can develop a definition of uncorrelatedness. Here we are only concerned with Boolean functions, and our definition of uncorrelatedness is somewhat simplified. Higher orders of uncorrelatedness have been defined by considering the subfunctions obtained by holding more than one input bit constant and the reader is referred to the papers by Siegenthaler [20, 21] and

Rueppel [18] for more details.

We introduce a geometrical representation of uncorrelated functions and prove several results concerning certain families of uncorrelated functions. By considering uncorrelated functions over infinite fields we also obtain a characterisation in terms of a vector space, and derive a general expression for its basis vectors. For other results concerning uncorrelatedness, especially the enumeration of functions satisfying combinations of criteria see [15].

2.2 Definitions

A function $f : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ is said to be uncorrelated with respect to x_i if and only if

$$\sum_{\underline{x} \epsilon \mathbf{Z}_2^n | x_i = 1} f(\underline{x}) = \sum_{\underline{x} \epsilon \mathbf{Z}_2^n | x_i = 0} f(\underline{x}). \tag{2.1}$$

These sums of values in $\mathbf{Z}_2$ are of course computed in $\mathbf{Z}$ as usual. From the definition we see that the pre-image of 1 splits into two sets of equal size, those vectors where $x_i = 1$ and those where $x_i = 0$.

If a function f is uncorrelated with respect to all $x_i : 1 \leq i \leq n$, then f is said to be uncorrelated.

2.3 Uncorrelated Hypercubes

Consider the map $\Phi : \mathbf{Z}_2^n \rightarrow \{1, -1\}^n$ defined by

$$\Phi(\underline{x}) = ((-1)^{x_1}, \ldots, (-1)^{x_n}). \tag{2.2}$$

Using Φ we map the domain of a function to the vertices of an n-dimensional hypercube, of side length 2, centred at the origin. At each vertex we can imagine a unit mass if the function evaluated at the point corresponding to the vertex is 1, and a zero mass otherwise.

It is easy to see that the condition that f be uncorrelated is precisely the condition that the centre of gravity of f's associated hypercube is at the origin. We will refer to such a hypercube as uncorrelated.

We now consider the problem of determining the number of uncorrelated functions by examining the number of uncorrelated hypercubes. Initially we consider the small cases $n = 1, 2, 3, 4$.

When $n = 1$ the domain is mapped to a line, and there are 2 uncorrelated solutions shown in Figure 2.1.

Figure 2.1: The 2 uncorrelated lines

When $n = 2$ the domain is mapped to a square, and there are 4 uncorrelated solutions shown in Figure 2.2.

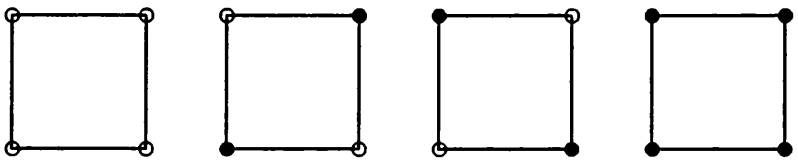

Figure 2.2: The 4 uncorrelated squares

When $n = 3$ the domain is mapped to a cube, and there are 18 uncorrelated solutions shown in Figure 2.3.

When $n = 4$ the domain is mapped to a hypercube, and there are 648 uncorrelated solutions.

Given an uncorrelated n-dimensional hypercube it immediately follows from the geometrical realisation that any rotation or reflection will yield another uncorrelated n-dimensional hypercube, as will simultaneously replacing all unit masses with a zero mass, and vice versa.

Siegenthaler [20] has given a simple algorithm for constructing uncorrelated functions for arbitrary sizes of n, satisfying any order of correlation from uncorrelated functions of a lower order. However, when translated to n-dimensional hypercubes, his solutions are only of the form of uncorrelated n-dimensional hypercubes constructed from two uncorrelated $(n-1)$-dimensional uncorrelated hypercubes, which in turn may be constructed from two $(n-2)$-dimensional uncorrelated hypercubes etc. This approach will always miss some uncorrelated functions. If there are k n-dimensional uncorrelated hypercubes, then there will be k^2 $(n+1)$-dimensional uncorrelated hypercubes constructible using this algorithm. For example, by combining the 4 uncorrelated functions in $\mathbf{Z}_2^2 \rightarrow \mathbf{Z}_2$ we can construct only 16 of the 18 uncorrelated functions in $\mathbf{Z}_2^3 \rightarrow \mathbf{Z}_2$, and for larger n an increasingly larger proportion of functions will be missed. The two uncorrelated functions missed by this algorithm are $x_1 + x_2 + x_3$ and $x_1 + x_2 + x_3 + 1$ and it is easy to see that such sum functions will be uncorrelated for all values of n.

2.4 An Upper Bound for the number of Uncorrelated Functions

Some further infinite families of uncorrelated hypercubes can be found by considering the sub-problem: How many uncorrelated n-dimensional hypercubes are there with precisely m (non-zero) masses? Trivially, any uncorrelated hypercube must have an even number of masses, and by symmetry we only need examine $m \leq 2^{n-1}$.

Let $\{\underline{v}_1, \ldots, \underline{v}_m\}$ be a set of column vectors representing the positions of the m masses in an uncorrelated hypercube. Construct an $m \times n$ table by listing these m column vectors.

It will have the form

$$\begin{array}{lcccccc} \underline{v}_1 & (& 1 & -1 & 1 & -1 & \ldots \;) \\ \underline{v}_2 & (& 1 & 1 & -1 & -1 & \ldots \;) \\ \vdots & & & & \vdots & & \\ \underline{v}_m & (& -1 & 1 & 1 & -1 & \ldots \;). \end{array}$$

The numbers in the table have been chosen at random to serve as an example.

This table has the property that each column will sum to 0, reflecting the fact that the centre of gravity of the masses is at the origin. Conversely, any $m \times n$ table of entries drawn from the set $\{1, -1\}$ with this property will correspond to an uncorrelated hypercube, provided that no two rows are the same. Two or more rows identical would imply more than one mass at a vertex which is disallowed.

Each uncorrelated hypercube will appear in $m!$ such tables, due to permutations of the rows. Therefore, the number of uncorrelated n-dimensional hypercubes with m masses will be the number of $m \times n$ tables with entries from $\{1, -1\}$, which satisfy the row and column criteria, divided by $m!$.

Let us introduce the notation $U_{n,m}$ for the number of uncorrelated n-dimensional hypercubes with m masses, or equivalently the number of uncorrelated functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ which take the value 1 precisely m times. Naturally U_n denotes the number of uncorrelated functions in $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$, and

$$U_n = \sum_{m \text{ even}}^{2^n} U_{n,m}. \tag{2.3}$$

We know that each column in our table consists of m values drawn from $\{1, -1\}$ which sum to 0. Therefore, there must be $(m/2)$ "1"s and $(m/2)$ "-1"s. This means that there are only $\binom{m}{m/2}$ possibilities for each column, and we immediately obtain an upper bound of $\frac{\binom{m}{m/2}^n}{m!}$ for the number of uncorrelated n-dimensional hypercubes with m masses, and an upper bound for U_n.

$$U_n \leq \sum_{m \text{ even}}^{2^n} \frac{\binom{m}{m/2}^n}{m!}. \tag{2.4}$$

2.5 Determination of the number of Uncorrelated Hypercubes with m masses, for small values of m

When $m = 2$ each column has $\binom{2}{1}$ possibilities, and the two rows cannot be the same. Therefore

$$U_{n,2} = \frac{\binom{2}{1}^n}{2!} = 2^{n-1}. \tag{2.5}$$

When $m = 4$ each column has $\binom{4}{2}$ possibilities. If a pair of rows is identical, then the two remaining rows must also be identical and there is no choice in their construction. We can pair off the rows in 3 ways, and having decided a pair of rows is to be identical we can construct it in 2^n ways, so there are 3×2^n tables disallowed. Therefore

$$U_{n,4} = \frac{\binom{4}{2}^n - 3 \times 2^n}{4!} = 2^{n-3}(3^{n-1} - 1). \tag{2.6}$$

When $m = 6$ each column has $\binom{6}{3}$ possibilities. Let us determine the number of tables with 2 or more rows the same. We can choose 2 rows to be identical in $\binom{6}{2} = 15$ ways, assign any of 2^n values to this pair, and complete each column in any one of 4 ways giving 15×8^n tables. However, this counts all those with 2 pairs of identical rows twice, and those with 2 triples of identical rows six times. A simple inclusion-exclusion argument yields

$$U_{n,6} = \frac{20^n - 15 \times 8^n + 45 \times 4^n - 40 \times 2^n}{720}. \tag{2.7}$$

Unfortunately, for $m \geq 8$ the same simple approach does not work. This is

because the number of ways that the columns may be completed, merely given that specified rows are identical, actually depends on what values appear in the specified rows. For example, given that rows 1 and 2 are identical, and rows 3 and 4 are identical, each column may be completed in either 1 or 6 ways, depending on the actual values occurring. This means that the simple inclusion-exclusion principle is not applicable to higher values of m.

The distribution of the uncorrelated hypercubes with respect to m is not uniform, with most uncorrelated hypercubes occurring around the central point $m = 2^{n-1}$.

n	$m = 0$	$m = 2$	$m = 4$	$m = 6$	$m = 8$
1	1	1	-	-	-
2	1	2	1	-	-
3	1	4	8	4	1
4	1	8	52	152	222
5	1	16	320	3824	?
6	1	32	1936	83168	?

Table 2.1: The number of uncorrelated hypercubes for some values of m and n

Table 2.1 gives some values of $U_{n,m}$. Those uncorrelated hypercubes where $m = 2^{n-1}$ are most important, since they represent uncorrelated functions which are also balanced. To illustrate the complexities of uncorrelated and balanced functions we investigate the case $n = 4$, $m = 8$ attempting to classify the solutions.

2.6 The Classification of Solutions for the case $n = 4,\ m = 8$

From Table 2.1, there are 222 uncorrelated hypercubes with 8 masses. These correspond to the 222 functions from $\mathbf{Z}_2^3 \rightarrow \mathbf{Z}_2$ which are both uncorrelated and balanced. There are three obvious ways to collect together solutions which are equivalent in some sense. We may regard as equivalent: firstly, any solutions obtained from each other by permuting the co-ordinate axes; secondly, any solutions obtained from each other by reflection about one or more axis; and thirdly, any solutions obtained from each other by complementing the mass at each vertex (that is, replacing a mass of value 1 by a mass of value 0 and vice versa). In functional terms, these 3 operations correspond to permuting the input variables, replacing x_i by $x_i + 1$, and adding 1 respectively (all operations performed modulo 2). The third has the simple result of pairing off all the solutions, reducing the problem by a factor of 2.

Using the group theoretic program CAYLEY [3], a group generated by the symmetry group of the 4 axes and the reflections about each axis acted on the 222 solutions. The solutions split into 7 orbits of respective sizes: $2, 8, 12, 24, 32, 48, 96$ as follows:

$$x_1 + x_2 + x_3 + x_4(+1)\} \quad \text{Orbit Size 2}$$

$$\left.\begin{array}{ll} x_1 + x_2 + x_3(+1) & x_1 + x_2 + x_4(+1) \\ x_1 + x_3 + x_4(+1) & x_2 + x_3 + x_4(+1) \end{array}\right\} \quad \text{Orbit Size 8}$$

$$\left.\begin{array}{lll} x_1 + x_2(+1) & x_1 + x_3(+1) & x_1 + x_4(+1) \\ x_2 + x_3(+1) & x_2 + x_4(+1) & x_3 + x_4(+1) \end{array}\right\} \quad \text{Orbit Size 12}$$

$x_1 + x_2 + x_3x_4(+1)$	$x_1 + x_2x_3 + x_4(+1)$	Orbit Size 48
$x_1 + x_2x_4 + x_3(+1)$	$x_1x_2 + x_3 + x_4(+1)$	
$x_1x_3 + x_2 + x_4(+1)$	$x_1x_4 + x_2 + x_3(+1)$	
$x_1 + x_2 + x_3x_4 + x_3(+1)$	$x_1 + x_2 + x_3x_4 + x_4(+1)$	
$x_1 + x_2x_3 + x_2 + x_4(+1)$	$x_1 + x_2x_3 + x_3 + x_4(+1)$	
$x_1 + x_2x_4 + x_2 + x_3(+1)$	$x_1 + x_2x_4 + x_3 + x_4(+1)$	
$x_1x_2 + x_1 + x_3 + x_4(+1)$	$x_1x_2 + x_2 + x_3 + x_4(+1)$	
$x_1x_3 + x_1 + x_2 + x_4(+1)$	$x_1x_3 + x_2 + x_3 + x_4(+1)$	
$x_1x_4 + x_1 + x_2 + x_3(+1)$	$x_1x_4 + x_2 + x_3 + x_4(+1)$	
$x_1 + x_2 + x_3x_4 + x_3 + x_4(+1)$	$x_1 + x_2x_3 + x_2 + x_3 + x_4(+1)$	
$x_1 + x_2x_4 + x_2 + x_3 + x_4(+1)$	$x_1x_2 + x_1 + x_2 + x_3 + x_4(+1)$	
$x_1x_3 + x_1 + x_2 + x_3 + x_4(+1)$	$x_1x_4 + x_1 + x_2 + x_3 + x_4(+1)$	

$x_1 + x_2x_3 + x_2x_4 + x_3x_4(+1)$

$x_1x_2 + x_1x_3 + x_2x_3 + x_4(+1)$

$x_1x_2 + x_1x_4 + x_2x_4 + x_3(+1)$

$x_1x_3 + x_1x_4 + x_2 + x_3x_4(+1)$

$x_1 + x_2x_3 + x_2x_4 + x_2 + x_3x_4 + x_3(+1)$

$x_1 + x_2x_3 + x_2x_4 + x_2 + x_3x_4 + x_4(+1)$

$x_1 + x_2x_3 + x_2x_4 + x_3x_4 + x_3 + x_4(+1)$

$x_1x_2 + x_1x_3 + x_1 + x_2x_3 + x_2 + x_4(+1)$

$x_1x_2 + x_1x_3 + x_1 + x_2x_3 + x_3 + x_4(+1)$

$x_1x_2 + x_1x_3 + x_2x_3 + x_2 + x_3 + x_4(+1)$

$x_1x_2 + x_1x_4 + x_1 + x_2x_4 + x_2 + x_3(+1)$

$x_1x_2 + x_1x_4 + x_1 + x_2x_4 + x_3 + x_4(+1)$

$x_1x_2 + x_1x_4 + x_2x_4 + x_2 + x_3 + x_4(+1)$

$x_1x_3 + x_1x_4 + x_1 + x_2 + x_3x_4 + x_3(+1)$

$x_1x_3 + x_1x_4 + x_1 + x_2 + x_3x_4 + x_4(+1)$

$x_1x_3 + x_1x_4 + x_2 + x_3x_4 + x_3 + x_4(+1)$

} Orbit Size 32

$$x_1x_2 + x_1x_3 + x_1 + x_2x_4 + x_2 + x_3x_4(+1)$$
$$x_1x_2 + x_1x_3 + x_1 + x_2x_4 + x_3x_4 + x_3(+1)$$
$$x_1x_2 + x_1x_3 + x_2x_4 + x_2 + x_3x_4 + x_4(+1)$$
$$x_1x_2 + x_1x_3 + x_2x_4 + x_3x_4 + x_3 + x_4(+1)$$
$$x_1x_2 + x_1x_4 + x_1 + x_2x_3 + x_2 + x_3x_4(+1)$$
$$x_1x_2 + x_1x_4 + x_1 + x_2x_3 + x_3x_4 + x_4(+1)$$
$$x_1x_2 + x_1x_4 + x_2x_3 + x_2 + x_3x_4 + x_3(+1)$$
$$x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_3 + x_4(+1)$$
$$x_1x_3 + x_1x_4 + x_1 + x_2x_3 + x_2x_4 + x_3(+1)$$
$$x_1x_3 + x_1x_4 + x_1 + x_2x_3 + x_2x_4 + x_4(+1)$$
$$x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2 + x_3(+1)$$
$$x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2 + x_4(+1)$$

} Orbit Size 24

$x_1 + x_2x_3 + x_2 + x_3x_4(+1)$	$x_1 + x_2x_3 + x_2x_4 + x_3(+1)$	
$x_1 + x_2x_3 + x_2x_4 + x_4(+1)$	$x_1 + x_2x_3 + x_3x_4 + x_4(+1)$	
$x_1 + x_2x_4 + x_2 + x_3x_4(+1)$	$x_1 + x_2x_4 + x_3x_4 + x_3(+1)$	
$x_1x_2 + x_1 + x_2x_3 + x_4(+1)$	$x_1x_2 + x_1 + x_2x_4 + x_3(+1)$	
$x_1x_2 + x_1x_3 + x_2 + x_4(+1)$	$x_1x_2 + x_1x_3 + x_3 + x_4(+1)$	
$x_1x_2 + x_1x_4 + x_2 + x_3(+1)$	$x_1x_2 + x_1x_4 + x_3 + x_4(+1)$	
$x_1x_2 + x_2x_3 + x_3 + x_4(+1)$	$x_1x_2 + x_2x_4 + x_3 + x_4(+1)$	
$x_1x_3 + x_1 + x_2 + x_3x_4(+1)$	$x_1x_3 + x_1 + x_2x_3 + x_4(+1)$	
$x_1x_3 + x_1x_4 + x_2 + x_3(+1)$	$x_1x_3 + x_1x_4 + x_2 + x_4(+1)$	
$x_1x_3 + x_2 + x_3x_4 + x_4(+1)$	$x_1x_3 + x_2x_3 + x_2 + x_4(+1)$	
$x_1x_4 + x_1 + x_2 + x_3x_4(+1)$	$x_1x_4 + x_1 + x_2x_4 + x_3(+1)$	
$x_1x_4 + x_2 + x_3x_4 + x_3(+1)$	$x_1x_4 + x_2x_4 + x_2 + x_3(+1)$	
$x_1 + x_2x_3 + x_2 + x_3x_4 + x_3(+1)$	$x_1 + x_2x_3 + x_2x_4 + x_2 + x_3(+1)$	Orbit Size 96
$x_1 + x_2x_3 + x_2x_4 + x_2 + x_4(+1)$	$x_1 + x_2x_3 + x_3x_4 + x_3 + x_4(+1)$	
$x_1 + x_2x_4 + x_2 + x_3x_4 + x_4(+1)$	$x_1 + x_2x_4 + x_3x_4 + x_3 + x_4(+1)$	
$x_1x_2 + x_1 + x_2x_3 + x_2 + x_4(+1)$	$x_1x_2 + x_1 + x_2x_4 + x_2 + x_3(+1)$	
$x_1x_2 + x_1x_3 + x_1 + x_2 + x_4(+1)$	$x_1x_2 + x_1x_3 + x_1 + x_3 + x_4(+1)$	
$x_1x_2 + x_1x_4 + x_1 + x_2 + x_3(+1)$	$x_1x_2 + x_1x_4 + x_1 + x_3 + x_4(+1)$	
$x_1x_2 + x_2x_3 + x_2 + x_3 + x_4(+1)$	$x_1x_2 + x_2x_4 + x_2 + x_3 + x_4(+1)$	
$x_1x_3 + x_1 + x_2 + x_3x_4 + x_3(+1)$	$x_1x_3 + x_1 + x_2x_3 + x_3 + x_4(+1)$	
$x_1x_3 + x_1x_4 + x_1 + x_2 + x_3(+1)$	$x_1x_3 + x_1x_4 + x_1 + x_2 + x_4(+1)$	
$x_1x_3 + x_2 + x_3x_4 + x_3 + x_4(+1)$	$x_1x_3 + x_2x_3 + x_2 + x_3 + x_4(+1)$	
$x_1x_4 + x_1 + x_2 + x_3x_4 + x_4(+1)$	$x_1x_4 + x_1 + x_2x_4 + x_3 + x_4(+1)$	
$x_1x_4 + x_2 + x_3x_4 + x_3 + x_4(+1)$	$x_1x_4 + x_2x_4 + x_2 + x_3 + x_4(+1)$	

An interesting result is obtained if the symmetry group alone (in other words,

no reflections about axes) is used to produce orbits. All orbits remain the same except the 24 element one, which splits into two halves, the mirror images of each other. These halves consist of the 12 functions which have a 1 in their polynomial form, and those that do not. These are the only solutions which cannot be superimposed (after a suitable rotation) on their complementary solutions.

2.7 Generalised Uncorrelated Functions

In our original formulation of the problem, we considered placing identical unit masses at the vertices of a n-dimensional hypercube. These solutions do not form a vector space over $\mathbf{Z}_2$, however, as when two such solutions are "added" together, the masses will be computed modulo 2, thus probably destroying the hypercube's balance.

Let us define a generalised uncorrelated function as a map $f : \mathbf{Z}_2^n \rightarrow \mathbf{K}$, which satisfies

$$\sum_{\underline{x}|x_i=1} f(\underline{x}) - \sum_{\underline{x}|x_i=0} f(\underline{x}) = 0 \quad \forall i : 1 \leq i \leq n, \tag{2.8}$$

where the sums are computed over $\mathbf{K}$ which is an an arbitrary field of characteristic zero.

The reason that we restrict $\mathbf{K}$ to fields of zero characteristic is to remove the phenomenon of masses "disappearing" when added over $\mathbf{K}$ which would otherwise prevent the set of generalised uncorrelated functions in $\mathbf{Z}_2^n \rightarrow \mathbf{K}$ forming a vector space. However, when $\mathbf{K}$ has characteristic zero the generalised uncorrelated functions do form a vector space over $\mathbf{K}$, and they can be elegantly characterised.

Theorem 2.1

When **K** is a field of zero characteristic, the generalised uncorrelated functions form a vector space over **K** of dimension $2^n - n$.

Proof

Let $f, g : \mathbf{Z}_2^n \rightarrow \mathbf{K}$ be generalised uncorrelated functions, and $\lambda, \mu \epsilon \mathbf{K}$, then

$$\sum_{\underline{x}|x_i=1} \lambda f + \mu g = \lambda \sum_{\underline{x}|x_i=1} f + \mu \sum_{\underline{x}|x_i=1} g$$

$$= \lambda \sum_{\underline{x}|x_i=0} f + \mu \sum_{\underline{x}|x_i=0} g = \sum_{\underline{x}|x_i=0} \lambda f + \mu g,$$

so the generalised uncorrelated functions form a vector space over **K**.

Let $\theta : (\mathbf{Z}_2^n \rightarrow \mathbf{K}) \rightarrow \mathbf{K}^n$ be defined by

$$\theta(f) = \Big(\sum_{\underline{x}|x_1=1} f(\underline{x}) - \sum_{\underline{x}|x_1=0} f(\underline{x}), \ldots, \sum_{\underline{x}|x_n=1} f(\underline{x}) - \sum_{\underline{x}|x_n=0} f(\underline{x}) \Big). \qquad (2.9)$$

It is obvious from the definition that θ satisfies $\theta(f+g) = \theta(f) + \theta(g)$, and $\theta(\lambda f) = \lambda\theta(f)$. So θ is a linear transformation acting on a finite dimensional vector space. The generalised uncorrelated functions are the kernel of θ and its image will be the whole of $\mathbf{K}^n$. To prove this, consider finding an f' such that $\theta(f') = (\alpha_1, \ldots, \alpha_n)$ for arbitrary $\alpha_1, \ldots, \alpha_n \epsilon \mathbf{K}$. Define

$$f'(\underline{x}) = \sum_{j=1}^{n} \frac{\alpha_j x_j}{2^{n-1}}, \qquad (2.10)$$

then the i-th component of $\theta(f')$ will be

$$\sum_{\underline{x}|x_i=1} f'(\underline{x}) - \sum_{\underline{x}|x_i=0} f'(\underline{x}) = \sum_{\underline{x}|x_i=1} \sum_{j=1}^{n} \frac{\alpha_j x_j}{2^{n-1}} - \sum_{\underline{x}|x_i=0} \sum_{j=1}^{n} \frac{\alpha_j x_j}{2^{n-1}}$$

$$= \sum_{\underline{x}|x_i=1} \left(\sum_{j=1,j\neq i}^{n} \frac{\alpha_j x_j}{2^{n-1}} + \frac{\alpha_i}{2^{n-1}} \right) - \sum_{\underline{x}|x_i=0} \left(\sum_{j=1,j\neq i}^{n} \frac{\alpha_j x_j}{2^{n-1}} \right) = \sum_{\underline{x}|x_i=1} \frac{\alpha_i}{2^{n-1}} = \alpha_i. \tag{2.11}$$

Therefore $\theta(f') = (\alpha_1, \ldots, \alpha_n)$ and the image of θ is the whole of $\mathbf{K}^n$. Using the "rank–nullity" equation, see [16] for example, we can find the dimension of the vector space of generalised uncorrelated functions.

$$\text{dimension (Kernel } \theta) + \text{dimension (Image } \theta) = \text{dimension } (\mathbf{Z}_2^n \rightarrow \mathbf{K}) \tag{2.12}$$

The dimension of the space of functions from $\mathbf{Z}_2^n \rightarrow \mathbf{K}$ is simply 2^n because each such function can be expressed in algebraic normal form as a sum of 2^n monomials over $\mathbf{K}$. Therefore,

$$\text{dimension (Kernel } \theta) + n = 2^n. \tag{2.13}$$

Thus the subspace of generalised uncorrelated functions has dimension $2^n - n$.

2.8 Finding a Basis for the Generalised Uncorrelated Functions

We know that the generalised uncorrelated functions form a vector space over $\mathbf{K}$ of dimension $2^n - n$. We now present an algorithm for explicity calculating a basis for this vector space.

Recall that a generalised uncorrelated function $f : \mathbf{Z}_2^n \rightarrow \mathbf{K}$ satisfies

$$\sum_{\underline{x}|x_i=1} f(\underline{x}) - \sum_{\underline{x}|x_i=0} f(\underline{x}) = 0 \quad \forall i : 1 \leq i \leq n.$$

For $0 \leq i < 2^n - 1$ we write y_i for $f(\underline{x})$ where $\underline{x}$ is the binary representation of i. Now we can restate the problem of finding a basis for the generalised

uncorrelated functions as the problem of finding a basis for the solution space of n simultaneous equations in 2^n unknowns. We seek the vector space consisting of all vectors of the form $(y_0, \ldots, y_{2^n-1})$ satisfying $\mathbf{A}_n \underline{y} = \mathbf{0}_n$ where

$$\mathbf{A}_n = \begin{pmatrix} 1 & 1 & \ldots & 1 & 1 & -1 & -1 & \ldots & -1 & -1 \\ 1 & 1 & \ldots & -1 & -1 & 1 & 1 & \ldots & -1 & -1 \\ & \vdots & & & & & & \vdots & & \\ 1 & 1 & -1 & -1 & \ldots & & \ldots & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & \ldots & & \ldots & 1 & -1 & 1 & -1 \end{pmatrix},$$

$$\underline{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{2^n-2} \\ y_{2^n-1} \end{pmatrix}, \quad \text{and } \mathbf{0}_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

The matrix $\mathbf{A}_n$ has n rows and 2^n columns. Each row consists of alternating sequences of 1's and -1's, starting with a sequence of 1's. The length of the sequences in a particular row is constant, and is half the length in the previous row, decreasing from 2^{n-1} for row 1 to 1 for row n. The column vector $\mathbf{0}_n$ is the $n \times 1$ matrix with all entries zero.

We know that the solution space has dimension $2^n - n$, so any n linearly independent y_i's must be expressible in terms of the remaining $2^n - n$. Picking a linearly independent n-subset is easy, as we can exploit the geometrical representation in terms of hypercubes. The variables $y_1, y_2, y_4, \ldots, y_{2^{n-1}}$ represent the masses at n vertices which lie in mutually perpendicular planes, and so are certainly lin-

early independent. Let $\mathbf{A}'_n$ be matrix $\mathbf{A}_n$ with columns $1, 2, 4, \ldots, 2^{n-1}$ deleted, and $\underline{y}'$ be column vector $\underline{y}$ with rows $1, 2, 4, \ldots, 2^{n-1}$ deleted. Re-arranging $\mathbf{A}_n \underline{y} = 0_n$, we obtain

$$\begin{pmatrix} 1 & 1 & 1 & \ldots & -1 \\ & & \vdots & & \\ 1 & 1 & -1 & \ldots & 1 \\ 1 & -1 & 1 & \ldots & 1 \\ -1 & 1 & 1 & \ldots & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_4 \\ \vdots \\ y_{2^{n-1}} \end{pmatrix} = \mathbf{B}_n \begin{pmatrix} y_1 \\ y_2 \\ y_4 \\ \vdots \\ y_{2^{n-1}} \end{pmatrix} = -\mathbf{A}'_n \underline{y}', \tag{2.14}$$

where matrix $\mathbf{B}_n$ is defined by

$$\mathbf{B}_n = (b_{ij}) = \begin{cases} -1 & \text{if } i + j = n + 1 \\ 1 & \text{otherwise} \end{cases} . \tag{2.15}$$

Therefore,

$$\begin{pmatrix} y_1 \\ y_2 \\ y_4 \\ \vdots \\ y_{2^{n-1}} \end{pmatrix} = -\mathbf{B}_n^{-1} \mathbf{A}'_n \begin{pmatrix} y_0 \\ y_3 \\ y_5 \\ \vdots \\ y_{2^n-1} \end{pmatrix} \tag{2.16}$$

Now we must calculate the inverse of matrix $\mathbf{B}_n$. Let the top row of $\mathbf{B}_n^{-1}$ be $(c_1, c_2, \ldots, c_n)$. By multiplying this row by each column of $\mathbf{B}_n$ in turn, we obtain the top row of the $n \times n$ identity matrix. Thus we deduce n equations in the n unknowns $c_1, c_2, \ldots, c_n$. When solved these give $c_i = \frac{1}{2(n-2)}$ for $0 \leq i \leq n-1$,

and $c_n = \frac{3-n}{2(n-2)}$. Repeating this procedure for each row, we deduce that

$$\mathbf{B}_n^{-1} = \frac{1}{2}\begin{pmatrix} \frac{1}{n-2} & \frac{1}{n-2} & \frac{1}{n-2} & \cdots & \frac{3-n}{n-2} \\ \frac{1}{n-2} & \frac{1}{n-2} & \frac{1}{n-2} & \cdots & \frac{1}{n-2} \\ \frac{1}{n-2} & \frac{1}{n-2} & \frac{3-n}{n-2} & \cdots & \frac{1}{n-2} \\ \frac{1}{n-2} & \frac{3-n}{n-2} & \frac{1}{n-2} & \cdots & \frac{1}{n-2} \\ \frac{3-n}{n-2} & \frac{1}{n-2} & \frac{1}{n-2} & \cdots & \frac{1}{n-2} \end{pmatrix}. \tag{2.17}$$

When (2.16) is multiplied out, it expresses $y_1, y_2, y_4, \ldots, y_{2^{n-1}}$ as linear combinations of $y_0, y_3, y_5, \ldots, y_{2^n-1}$, the remaining y_is. For $i = 1, 2, 4, \ldots, 2^{n-1}$, let

$$y_i = \sum_{y_j \in \underline{y}'} k_{i,j} y_j. \tag{2.18}$$

Therefore we can deduce:

Theorem 2.2

The $2^n - n$ vectors

$$\begin{array}{cccccccccc} (& 1, & k_{1,0}, & k_{2,0}, & 0, & k_{4,0}, & \ldots & k_{2^{n-1},0}, & \ldots & 0 \;) \\ (& 0, & k_{1,3}, & k_{2,3}, & 1, & k_{4,3}, & \ldots & k_{2^{n-1},3}, & \ldots & 0 \;) \\ (& 0, & k_{1,5}, & k_{2,5}, & 0, & k_{4,5}, & \ldots & k_{2^{n-1},5}, & \ldots & 0 \;) \\ & & & & & \vdots & & & & \\ (& 0, & k_{1,2^n-1}, & k_{2,2^n-1}, & 0, & k_{4,2^n-1}, & \ldots & k_{2^{n-1},2^n-1}, & \ldots & 1 \;) \end{array}$$

form a basis for the vector space of generalised uncorrelated functions over $\mathbf{K}$.

Note that the basis is just the rows of the $(2^n - n) \times (2^n - n)$ identity matrix with n additional columns inserted. If we number our columns from zero, then these additional columns appear at positions $1, 2, 4, 8, \ldots, 2^{n-1}$. The entries in

the column at position i consist of the coefficients of y_i when expressed in terms of our basis vectors.

The proof is largely contained in the preceeding derivations, but we should note that if any of the rows were linearly dependent on the others, we could obtain a lower dimension than $2^n - n$ for the space of generalised uncorrelated functions, a contradiction. Therefore these $2^n - n$ vectors must generate the space.

The above basis derivation is far simpler to use than its appearance suggests. It is illustrated here with the case $n = 3$.

Our matrix equation for $n = 3$ is

$$\begin{pmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \qquad (2.19)$$

Now we select y_1, y_2, y_4 as our linearly independent vectors,

$$\begin{pmatrix} y_1 \\ y_2 \\ y_4 \end{pmatrix} = - \begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_3 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} \right]$$

$$= \begin{pmatrix} -1 & 0 & 0 & 1 & 1 \\ -1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_3 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix}. \tag{2.20}$$

So we have expressed y_1, y_2 and y_4 in terms of our basis vectors. Now we insert these 3 rows as column vectors into the 5×5 identity matrix, in positions 1, 2 and 4 to obtain the matrix

$$\begin{pmatrix} 1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

The 5 basis vectors are the simply the rows of this matrix.

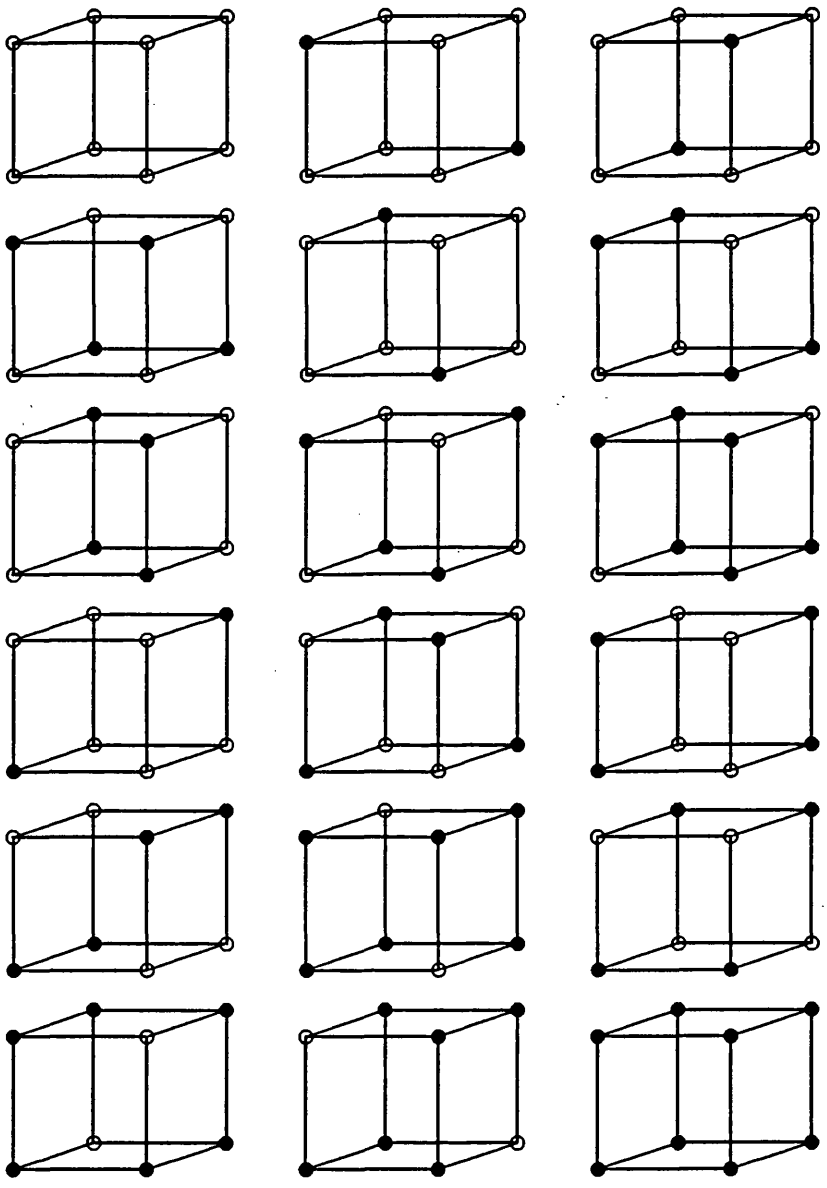

Figure 2.3: The 18 uncorrelated cubes

Chapter 3

Computationally Equivalent Functions and the DES

3.1 Introduction

Consider two functions $f(x, y, z) = xy + z$ and $g(x, y, z) = x + yz$. The idea of computationally equivalent functions is motivated by the observation that f and g can be computed by the same hardware implementation. Given a box which computes $f(x, y, z)$, we can use it to also compute $g(x, y, z)$.

$f(x, y, z)$ $g(x, y, z)$

This works because $g(x, y, z) = f(\pi(x, y, z))$ where π is a permutation of the input bits. If $g(x, y, z)$ can be computed from $f(x, y, z)$ in this way, f and g are equivalent in some sense. This equivalence is not particularly useful, however, since only a few functions will be comparable. A more realistic definition allows the complementing of input bits, as well as their permutation.

By defining a group of permutations and complements we consider partitioning the polynomials of functions into equivalence classes. We derive results concerning the sizes of the classes and the probability of two functions being computationally equivalent. Exploiting the factorisation of the group we develop an algorithm for determining whether two functions are computationally equivalent, far faster than the brute force approach.

We then turn our attention to the Data Encryption Standard and use our algorithm to search for subtle relationships within the S-boxes. We also analyse the S-boxes for uncorrelation properties discussed in Chapter 2 and draw attention to an anomaly in one of the permutations used in the DES algorithm.

3.2 Definitions

For $1 \leq i \leq n$, define the i-th component of $\lambda_I : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2^n$ by

$$\lambda_I(x_1, \ldots, x_n)_i = \begin{cases} x_i + 1 & \text{if } i \in I \\ x_i & \text{otherwise.} \end{cases}$$

So λ_I simply complements those inputs specified in the set I.

Let $\Lambda = \{\lambda_I | I \subseteq [1, \ldots, n]\}$.

Let $f, g : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$, then $g \sim f$ if and only if

$$g(x_1, \ldots, x_n) = f(\lambda(\pi(x_1, \ldots, x_n))),$$

where π is a permutation of the n input bits, and $\lambda \in \Lambda$.

3.3 The group of permutations and complements

Theorem 3.1

Let $P_n = \{\pi \circ \lambda_I | \pi \in S_n, I \subseteq [1 \ldots n]\}$, where S_n denotes the symmetric group on n objects, then P_n is a group.

Here $(x_1, \ldots, x_n)(\pi \circ \lambda_I)$ is defined to be $\lambda_I(\pi(x_1, \ldots, x_n))$ so $\circ$ simply denotes function composition. We write the elements of P_n on the right-hand side of their arguments because we want $(\pi \circ \lambda_I)(\rho \circ \lambda_J)$ to mean the application of $(\pi \circ \lambda_I)$ followed by the application of $(\rho \circ \lambda_J)$ and it is more convenient to

associate starting at the left-hand side. In other words,

$$(x_1, \ldots x_n)(\pi \circ \lambda_I)(\rho \circ \lambda_J) = ((x_1, \ldots, x_n)(\pi \circ \lambda_I))(\rho \circ \lambda_J)$$

Proof

The identity of P_n is $\iota \circ \lambda_\emptyset$, where ι is the identity permutation.

For $\pi \circ \lambda_I, \rho \circ \lambda_J \in P_n$,

$$(x_1, \ldots, x_n)(\pi \circ \lambda_I)(\rho \circ \lambda_J) = \lambda_I(x_{\pi(1)}, \ldots, x_{\pi(n)})\rho \circ \lambda_J$$

$$= \lambda_J(\lambda_{\rho(I)}(x_{\rho(\pi(1))} \cdots x_{\rho(\pi(n))})) = \lambda_{J+\rho(I)}(x_{\rho(\pi(1))}, \ldots, x_{\rho(\pi(n))})$$

$$= (x_1, \ldots, x_n)(\pi \circ \rho) \circ \lambda_{J+\rho(I)},$$

so P_n is closed under composition. The expression $J + \rho(I)$ refers to exclusive set union, that is, $(J \cup \rho(I)) - (J \cap \rho(I))$.

$$(x_1, \ldots, x_n)(\pi \circ \lambda_I)(\pi^{-1}\lambda_{\pi^{-1}(I)}) = \lambda_I(x_{\pi(1)}, \ldots, x_{\pi(n)})\pi^{-1} \circ \lambda_{\pi^{-1}(I)}$$

$$= \lambda_{\pi^{-1}(I)}\lambda_{\pi^{-1}(I)}(x_1, \ldots, x_n) = (x_1, \ldots, x_n).$$

Thus, $(\pi \circ \lambda_I)^{-1} = \pi^{-1} \circ \lambda_{\pi^{-1}(I)}$, so inverses are present.

Each element of P_n consists of a permutation of the n input bits, and a member of Λ, which will complement some selection of the bits. In fact the group P_n is a wreath product. For more details on wreath products see [8].

We can now easily demonstrate that $\sim$ defined in the previous section is in fact an equivalence relation. We have

$$f(\lambda_0(\iota(x_1, \ldots, x_n))) = f(x_1, \ldots, x_n),$$

therefore $f \sim f$.

$$\text{If} \quad g(x_1, \ldots, x_n) = f(\lambda_I(\pi(x_1, \ldots, x_n)))$$

$$\text{then} \quad f(x_1, \ldots, x_n) = g(\lambda_{\pi^{-1}(I)}(\pi^{-1}(x_1, \ldots, x_n))),$$

so $f \sim g$ implies that $g \sim f$, and similarly $f \sim g, g \sim h$ implies that $f \sim h$ since groups are closed under composition.

3.4 Classification of $Z_2^n \rightarrow Z_2$ under the Action of P_n

Representing functions from $Z_2^n \rightarrow Z_2$ as polynomials in n variables over Z_2 as before, we can divide the 2^{2^n} functions into disjoint equivalence classes under the equivalence relation $\sim$. These are classes of computationally equivalent functions.

In order to calculate the equivalence classes for each n, we use the group theoretic program CAYLEY [3]. By generating the group P_n and allowing it to act on the polynomials representing each function, we can compute the orbits of the polynomials. These orbits will consist of computationally equivalent functions.

The algorithm has the following form:

Step 0: X is the set of all polynomials.

Step 1: Select x in X.

Step 2: Compute the orbit of x under P_n.

Step 3: Delete this orbit from X, X = X - x P_n.

The CAYLEY code for this algorithm can be found in Appendix A.

When $n = 1$, there are $2^{2^1} = 4$ polynomials which split into 3 equivalence classes: $\{0\}$, $\{1\}$, and $\{x_1, x_1 + 1\}$.

When $n = 2$, there are $2^{2^2} = 16$ polynomials which split into 6 equivalence classes:

$\{0\}$

$\{1\}$

$\{x_1, x_2, x_1 + 1, x_2 + 1\}$

$\{x_1 + x_2, x_1 + x_2 + 1\}$

$\{x_1x_2, x_1x_2 + x_1, x_1x_2 + x_2, x_1x_2 + x_1 + x_2\}$

$\{x_1x_2 + 1, x_1x_2 + x_1 + 1, x_1x_2 + x_2 + 1, x_1x_2 + x_1 + x_2 + 1\}$.

When $n = 3$, there are $2^{2^3} = 256$ polynomials which split into 22 equivalence classes listed in Table 3.1. Only one representative of each equivalence class is given, together with the size of the class.

The distributions of class sizes for the 65536 polynomials when $n = 4$ are given in Table 3.2.

3.5 Some Results Concerning Orbit Sizes

The stabiliser, $stab(f)$, of a polynomial f is defined by

$$stab(f) = \{\rho \in P_n | f\rho = f\}. \tag{3.1}$$

In other words, it is the set of all maps which leave f fixed. In fact $stab(f)$ is a subgroup of P_n, as can be trivially checked.

By the Orbit–Stabiliser Theorem (see [23] for details) we have

$$\#\left(\frac{P_n}{stab(f)}\right) = \#orbit(f). \tag{3.2}$$

From (3.2) we deduce that

$$\#orbit(f) \times \#stab(f) = n!2^n, \tag{3.3}$$

and therefore all orbit sizes divide $n!2^n$.

Theorem 3.2

For all $n \geq 1$, there are precisely 2 orbits of size 1, namely $\{0\}$, $\{1\}$.

Proof

The polynomials **0**, **1** are certainly mapped to themselves by all elements of P_n. We must prove that no other polynomials are invariant under P_n.

Let f be a function from $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$, such that f is not identically equal to **0** or **1**. Pick a variable x_k occurring in f, and consider f', the function obtained from f by replacing x_k by $x_k + 1$. This function is certainly in f's orbit, and it is not equal to f since $f \equiv f'$ implies $\frac{\partial f}{\partial x_k} = 0$, contradicting the fact that f depends on x_k. Therefore, f is not invariant under P_n.

Theorem 3.3

For all $n \geq 1$, there will be precisely 1 orbit of size 2, and this orbit will be

$\{\sum_{i=1}^{n} x_i, 1 + \sum_{i=1}^{n} x_i\}$.

Proof

It is easily checked that the polynomials $\sum_{i=1}^{n} x_i$ and $1 + \sum_{i=1}^{n} x_i$ are mapped to themselves, or each other, by all elements of P_n. Let f be a polynomial in an orbit of size 2. Pick a variable x_k occurring in f. Now, as in the proof of Theorem 3.2, consider f' formed by complementing x_k in f. These two polynomials f and f' are not equal, so must be the two members of the orbit.

We know that f and f' must contain all variables x_i, for $1 \leq i \leq n$ since if a variable were absent, x_l say, we could pick an element of P_n which mapped x_k to x_l, thereby generating a member of the orbit of f which was not equal to f or f'.

If $n = 1$ then the result holds trivially, otherwise pick an arbitrary variable x_m and let f'' be formed from f by complementing x_m. Now f' and f'' are both not equal to f, therefore they must be equal to each other. So, $f - f' = f - f''$, that is, $\frac{\partial f}{\partial x_k} = \frac{\partial f}{\partial x_m}$. The right-hand side does not contain x_m, so no terms containing x_k and x_m can be present in f. However, x_m was arbitrarily chosen, therefore the variables can only occur as terms of weight 1. Since all variables are known to be present we have $f = \sum_{i=1}^{n} x_i$ and $f' = 1 + \sum_{i=1}^{n} x_i$, or vice versa.

Theorem 3.4

For $k = 0, \ldots, n$ there is at least one orbit of size $\binom{n}{k} 2^k$ for each $n \geq 1$.

Proof

Consider the orbit of $x_1 \ldots x_k$. Each term in the orbit will consist of k variables chosen from a possible n, with none, some, or all variables then complemented. The total number is

$$\sum_{l=0}^{k} \binom{n}{k}\binom{k}{l} = \binom{n}{k}2^k. \tag{3.4}$$

Similarly, by considering the orbit of $x_1 + \ldots + x_k$, we obtain: for $k = 0, \ldots, n$ there is at least one orbit of size $2\binom{n}{k}$.

3.6 Calculation of the probability of two polynomials being computationally equivalent

If the 2^{2^n} polynomials have been partitioned into orbits under P_n, then we can derive an exact value for the probability that two randomly chosen polynomials, f and g say, lie in the same orbit.

Suppose that there are t orbits of sizes $o_1, \ldots, o_t$ respectively, where

$$\sum_{i=1}^{t} o_i = 2^{2^n}. \tag{3.5}$$

Then the probability that f and g both lie in the k-th orbit is

$$\left[\frac{o_k}{2^{2^n}}\right]^2.$$

So the probability that they lie in the same orbit is

$$\sum_{k=1}^{t} \left[\frac{o_k}{2^{2^n}}\right]^2 = 2^{-2^{n+1}} \sum_{k=1}^{p} o_k^2. \tag{3.6}$$

For $n = 1, 2, 3, 4$ the orbits have been computed so we can calculate the probabilities precisely. They are given in Table 3.3.

The maximum size possible for an orbit is $n!2^n$, therefore

$$\sum o_k^2 \leq \frac{2^{2^n}}{n!2^n}[n!2^n]^2 = 2^{2^n} n!2^n. \tag{3.7}$$

This gives us an upper bound on the probability that f and g lie in the same orbit:

$$\mathrm{P}[f \text{ and } g \text{ lie in same orbit}] \leq \frac{n!2^n}{2^{2^n}}. \tag{3.8}$$

3.7 The Decision Problem for Computationally Equivalent Functions

3.7.1 Finding an Invariant Property Under Permutation

Given a pair of functions $f, g : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ we would like an algorithm to determine whether $f \sim g$, that is, do they lie in the same orbit under the action of P_n. The brute force solution is to consider f under the action of each $\rho \in P_n$, and see whether g is produced. This has time complexity $n!2^n$, so is only practical for small n. We can do better by utilising the factorisation of P_n into two components, and finding a property which is invariant under just one of the components.

If f and g are computationally equivalent then $f(\pi \circ \lambda_I) = g$ for some $\pi \in P_n, \lambda_I \in \Lambda$. Therefore, $f\pi = g(\lambda_I^{-1}) = g\lambda_I$. Now consider f and g written out as polynomials in n variables over $\mathbf{Z}_2$. By the shape of the polynomial we simply mean the number of terms of each weight present. This property is invariant under a permutation of the variables. Therefore $f\pi$ and f have the same shape, so rather than try all of the $n!2^n$ possibilities for $\rho \in P_n$,

we need only try those of the form $\pi \circ \lambda_I$, where $g\lambda_I$ is the same shape as f. There are only 2^n possibilities for λ_I, so the time complexity is reduced to $(2^n + (\text{Expected number of shape matches} \times n!))$. We will now show that the probability of two polynomials having the same shape is small.

3.7.2 The Shape of a Polynomial

Polynomials representing functions from $\mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ naturally form a vector space over $\mathbf{Z}_2$. Picking the basis $\{1, x_1, \ldots, x_n, x_1x_2, \ldots, x_{n-1}x_n, \ldots, x_1 \ldots x_n\}$ we can write each polynomial as a sum of monomials in this basis. The basis consists of one term of weight 0, n terms of weight 1, and, in general, $\binom{n}{w}$ terms of weight w $(0 \leq w \leq n)$. We can formally define the shape of a polynomial to be sequence $< s_0, s_1, \ldots, s_n >$ where s_i denotes the number of basis terms of weight i present in the polynomial.

Each s_i can range from 0 to $\binom{n}{i}$ independently, so we immediately obtain an expression for the number of possible shapes for a polynomial in n variables:

$$\prod_{w=0}^{n} \left[\binom{n}{w} + 1 \right].$$

In order to find the probability of two polynomials having the same shape, we must calculate the probability distribution of the shape sequence. The probability that a function has $s_w = k$, that is, precisely k terms of weight w is

$$P[s_w = k] = \frac{\binom{\binom{n}{w}}{k}}{2^{\binom{n}{w}}} \quad \text{for } 0 \leq k \leq \binom{n}{w}. \tag{3.9}$$

Since, of the $2^{\binom{n}{w}}$ subsets of the $\binom{n}{w}$ terms of weight w, precisely $\binom{\binom{n}{w}}{k}$ will have k terms.

So the probability that two polynomials will have the same number of terms of weight w, for some fixed w is given by,

$$\sum_{k=0}^{\binom{n}{w}} \left[\frac{\binom{\binom{n}{w}}{k}}{2^{\binom{n}{w}}} \right]^2 = \frac{\sum_{k=0}^{N} \binom{N}{k}^2}{2^{2N}} = \frac{\binom{2N}{N}}{2^{2N}} \text{ where } N = \binom{n}{w}. \tag{3.10}$$

Therefore, given two shapes $s =< s_0, \ldots, s_n >$ and $t =< t_0, \ldots, t_n >$, the probability that the shapes are identical is simply the product of the probabilities of them agreeing at each element, namely

$$P[s = t] = \prod_{w=0}^{n} \frac{\binom{2N}{N}}{2^{2N}}, \text{ where } N = \binom{n}{w}$$

$$= \frac{\prod_{w=0}^{n} \binom{2\binom{n}{w}}{\binom{n}{w}}}{2^{\sum_{w=0}^{n} 2\binom{n}{w}}} = \frac{\prod_{w=0}^{n} \binom{2\binom{n}{w}}{\binom{n}{w}}}{2^{2^{n+1}}}. \tag{3.11}$$

In order to see the magnitude of this quantity we can use Stirling's approximation for $n!$ to obtain

$$2^{-2^{n+1}} \prod_{w=0}^{n} \binom{2\binom{n}{w}}{\binom{n}{w}} \approx 2^{-2^{n+1}} \prod_{w=0}^{n} \frac{2^{2\binom{n}{w}}}{\sqrt{\pi \binom{n}{w}}} = \frac{1}{\prod_{w=0}^{n} \sqrt{\pi \binom{n}{w}}} < \frac{1}{\sqrt{\pi^{n+1} n^{n-1}}}. \tag{3.12}$$

Let us say that two shapes are compatible if their last non-zero terms are equal in position and value (or if they both have no non-zero terms). In other words, $s = < s_0, \ldots, s_n >$ and $t = < t_0, \ldots, t_n >$ are compatible if $s_l = t_l > 0$ and $s_m = t_m = 0, \forall m : l < m \leq n$ for some $l \geq 0$ (or if s and t are both sequences of zeros). Now observe that a map of the form λ_I acting on a polynomial does not affect the last non-zero term, and the subsequent zero terms, of the polynomial's shape. Therefore, if we are trying to decide if $f \sim g$, and their shapes are not compatible, then they cannot be computationally equivalent.

What is the probability that two shapes are compatible? Let p_l be the probability that s and t are compatible and their last non-zero term is in position l.

Then the probability that s and t are compatible is clearly

$$\mathrm{P}(s \text{ and } t \text{ are sequences of zeros}) + \sum_{l=0}^{n} p_l = \frac{1}{2^{2n+1}} + \sum_{l=0}^{n} p_l. \tag{3.13}$$

Therefore,

$$\begin{aligned} p_l &= P[s_l = t_l > 0]. \prod_{j=l+1}^{n} P[s_j = t_j = 0] \\ &= \frac{\binom{2\binom{n}{l}}{\binom{n}{l}} - 1}{2^{2\binom{n}{l}}} \cdot \prod_{j=l+1}^{n} \frac{1}{2^{2\binom{n}{j}}} \\ &= \frac{\binom{2\binom{n}{l}}{\binom{n}{l}} - 1}{2^{2\binom{n}{l}}} \cdot \frac{1}{2^{2\sum_{j=l+1}^{n}\binom{n}{j}}}. \end{aligned} \tag{3.14}$$

From the definition, it is clear that $p_0 < p_1 < \ldots < p_{n-1} < p_n$ and

$$p_{n-1} = \frac{\binom{2n}{n} - 1}{2^{2n}} \cdot \frac{1}{2^2} \approx \frac{1}{4\sqrt{\pi n}} \quad \text{by Stirling's approximation.}$$

The presence of the second factor in the expression for p_l, when $l < n$, means that the p_l's rapidly tend to zero as n increases. So $\sum p_l$ is dominated by the term $p_n = \frac{1}{4}$. This corresponds to the simplest way two shapes can be compatible – by both having a term of weight n.

3.7.3 The Time Complexity of the New Algorithm

We can now estimate the expected number of shape matches in the algorithm. Given f and g we compute the 2^n polynomials $g\lambda_I$ and look for shape matches. If the shapes of f and g are not compatible, then there will not be a match, whereas if they are, then we increase the probability of a match by approximately 2^n. The 2^n polynomials are not an independent sample, indeed some of them

may be the same, so the factor is not precisely 2^n, but certainly bounded above by 2^n. The expected number of shape matches is not greater than

$$\frac{1}{4}.2^n.\frac{\prod_{w=0}^{n}\binom{2\binom{n}{w}}{\binom{n}{w}}}{2^{2^n+1}}.$$

So we have reduced the average running time from $n!2^n$ to approximately

$$\frac{n!2^{n-2}}{\prod_{w=0}^{n}\sqrt{\pi\binom{n}{w}}}.$$

We know that $\pi^{\frac{n+1}{2}} > 2^{\frac{n}{2}-2}$ and $\prod_{w=0}^{n}\binom{n}{w} > n!$, therefore

$$\text{Time complexity} \approx \frac{n!2^{n-2}}{\prod_{w=0}^{n}\sqrt{\pi\binom{n}{w}}} < \sqrt{n!2^n}. \tag{3.15}$$

3.8 Other invariant properties of functions under P_n

Several properties of binary functions previously studied are invariant under the action of P_n on their polynomials as well as uncorrelatedness.

Theorem 3.5

If $f, g : \mathbf{Z}_2^n \rightarrow \mathbf{Z}_2$ and $f \sim g$ then

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} f(\underline{x}) = \sum_{\underline{x} \in \mathbf{Z}_2^n} g(\underline{x}).$$

Proof

$$\sum_{\underline{x} \in \mathbf{Z}_2^n} g(\underline{x}) = \sum_{\underline{x} \in \mathbf{Z}_2^n} f(\lambda(\pi(\underline{x}))) \quad \text{for some } \lambda \in \Lambda, \pi \in S_n$$

$$= \sum_{\underline{x} \in Z_2^n} f(\lambda(\underline{x})) = \sum_{\underline{x} \in Z_2^n} f(\underline{x}),$$

since $\pi(\underline{x})$ and $\lambda(\underline{x})$ will also range over Z_2^n .

Corollary

Only a balanced function can be equivalent under P_n with its complement.

Theorem 3.6

If $f, g : Z_2^n \rightarrow Z_2$ and $f \sim g$ and f satisfies the Strict Avalanche Criterion (SAC) then g satisfies the SAC.

Proof

Recall from Chapter 1 that a function satisfies the SAC if, and only if, all its partial derivatives are balanced.

Let $g = f\pi \circ \lambda$ for some $\pi \circ \lambda \in P_n$,

$$\text{then } \frac{\partial g}{\partial x_i} = g(\underline{x} \oplus c_i) \oplus g(\underline{x}) = (f(\underline{x} \oplus c_i)\pi)\lambda \oplus (f(\underline{x})\pi)\lambda$$

$$= (f(\pi(\underline{x}) \oplus c_{\pi(i)}) \oplus f(\pi(\underline{x})))\lambda = (\frac{\partial f}{\partial x_{\pi(i)}})\pi \circ \lambda.$$

$$\text{Now } \frac{\partial g}{\partial x_i} \sim \frac{\partial f}{\partial x_{\pi(i)}} \quad \Rightarrow \quad \sum_{\underline{x} \in Z_2^n} \frac{\partial g}{\partial x_i} = \sum_{\underline{x} \in Z_2^n} \frac{\partial f}{\partial x_{\pi(i)}} = 2^{n-1}.$$

So g also satisfies the SAC. Similarly, if f satisfies the SAC of order m, then so does g.

3.9 Application of Equivalence Testing Algorithm to the DES S-boxes

At the heart of the Data Encryption Standard (DES) are the eight S-boxes. The S-boxes provide the only non-linear operation of the DES cipher and are thus essentially the only part of cryptologic importance. The S-boxes have naturally been the subject of much investigation, see for example [2] and [19].

Each S-box returns a 4-bit block when given a 6-bit block as input. The S-boxes can be written as 4×16 size tables where each row is a permutation of the numbers $0, \ldots, 15$. Given a 6-bit block $b_1 b_2 b_3 b_4 b_5 b_6$, the outermost bits, b_1 and b_6, determine a row, and the innermost bits determine a column. The 4-bit number thus specified is returned. For further details, and a discussion of the DES cipher, see [5]. Here we are only concerned with the functions making up the S-boxes.

By considering each row separately, we may regard an S-box as a collection of four functions from $\mathbf{Z}_2^4 \rightarrow \mathbf{Z}_2^4$. If we now take each output bit as an independent function, then the eight S-boxes yield 128 functions from $\mathbf{Z}_2^4 \rightarrow \mathbf{Z}_2$. All these 128 functions are balanced; that is, they take the values 0 and 1 equally often. This follows immediately from each row of an S-box consisting of the numbers $0, \ldots, 15$. Let us refer to these 128 functions as the S-box functions in the following sections.

These 128 functions are not all different. Let us call two identical functions a pair of identical twins, and two functions which are the binary complement of each other, fraternal twins. There are 10 pairs of identical twins, 9 pairs of

fraternal twins, and three sets of triplets, each consisting of a pair of identical twins and a fraternal sibling.

Each row of Table 3.5 specifies the positions of the identical twins and gives the function as a 16-bit string. This string is simply a listing of the function's output on the successive inputs $0000, 0001, 0010, \ldots, 1111$. The 4 output bits of the S-boxes have been numbered $1, 2, 3, 4$ for identification, bit 1 being the most significant (leftmost). Similarly for Table 3.6, with the function description being that of the first entry, and Table 3.7, where the third entry is the fraternal twin of the other two. Note that in [10] the rows of an S-box are numbered 1, 2, 3, 4. However, since they are selected by bits $b_1 b_6$ being respectively $00, 01, 10, 11$, the numbering 0, 1, 2, 3 has been preferred as more logical.

The identical and fraternal twins occurring between rows of the same S-box are given in [10], but those occurring between rows in different S-boxes are not. Furthermore the identical twins bit 1 of row 0 and bit 2 of row 2 in S-box 7 are omitted. The two members of each triplet occurring in the same S-box are given, but the fact they have a third sibling elsewhere is not mentioned. Clearly S-box 4 has an exceptionally high degree of structure, and in fact, as noted in [10], it is 75% redundant.

In an attempt to discover more subtle relationships within the S-boxes, the equivalence testing algorithm was implemented in the computer algebra system REDUCE [9] and the group theoretic program CAYLEY [3] and run on the 128 S-box functions and their complements (with duplicates removed). Each of the 206 remaining functions generated 16 derived functions corresponding to the 16 possibilities for input bit complementation. The 3296 derived functions then each had their shapes calculated. The shapes were sorted, and each

pair of derived functions with the same shape was tested for equivalence by applying the 24 possible permutations to one, and comparing with the second. The result of this large computation (running time approximately 12 days) was disappointingly negative:

Theorem 3.7

There are no computationally equivalent functions among the 128 S-box functions and their complements, apart from those which are equal (the twins and triplets).

This result does not of course rule out the possibility of further simple relationships being discovered within the S-boxes, but such relationships would be more complex than straightforward computational equivalence.

3.10 Bias in the S-box Functions

Another test applied to the 128 S-box functions was an analysis of how uncorrelated they were. The functions would be expected to show a high degree of uncorrelatedness. For each function the sums

$$|\sum_{x_i=1} f(\underline{x}) - \sum_{x_i=0} f(\underline{x})| \quad \text{for } i = 1,2,3,4$$

were computed. Let us refer to these values as the functions' biases with respect to each input bit. If a function is uncorrelated with respect to x_i then it will have zero bias with respect to x_i. The incidence of each bias is shown in Table 3.8. It follows from the definition that all biases must be even, and range from 0 to 8

in the case of functions from $\mathbf{Z}_2^4 \rightarrow \mathbf{Z}_2$. A bias of 8 would imply that a function was of the form $x_i(+1)$ so it is not surprising that no such bias occurred. More interestingly, no bias more than 2 was found. The only observable anomaly is more biases of 2 with respect to x_2, but the difference is not significant. Obviously there is a strong tendency towards uncorrelatedness. We can also look at the distribution when the biases are grouped into fours, being the biases with respect to each input bit derived for a particular S-box function. There are 512 biases, 333 of which are 0. If they were distributed randomly among the 128 S-box functions we would expect a simple binomial distribution with parameters 4 and $\frac{333}{512}$. Table 3.9 compares the predicted distribution with the actual numbers of S-box functions with respectively 0, 1, 2, 3, and 4 biases of 0. The missing 0.1 in the total of predicted frequencies is of course due to rounding. There is a reasonable fit between the predicted and observed frequencies and nothing to contradict the assumption that the biases are distributed randomly. There are only 2 functions which have no biases of 0, in other words they are not uncorrelated with respect to any of their input bits. Their locations are S-box 5, row 0, bit 3 and S-box 5, row 3, bit 1 but apart from occurring in the same S-box there is nothing suspicious about them. There are also 15 functions with four biases of 0, in other words uncorrelated functions. This is a density of 9.4% compared to a density of only 1.7% among balanced functions in $\mathbf{Z}_2^4 \rightarrow \mathbf{Z}_2$, confirming that the S-box functions exhibit a strong degree of uncorrelatedness.

3.11 An Observation Concerning the Inputs to the S-boxes

During analysis of the DES algorithm an undesirable feature of the permutation PC-2 was found. In the algorithm, a 32-bit block is expanded to a 48-bit block by cloning half its bits. This 48-bit block is then XORed with a 48-bit key derived from the original key, and the result passed to the S-boxes. Clearly, cloning bits increases the redundancy of the ciphertext and the key bits must remove any tendency for the cloned bits to remain equal. The positions of the cloned bits appear in a regular pattern, and by computing the derived key, we can see which bits of the key are XORed with each pair of cloned bits per iteration. This approach was used in [6] where the consequences of holding certain input bits to the S-boxes constant were analysed. Ideally, each pair of cloned bits would be XORed with 2 different bits of the key on each iteration. Most pairs of cloned bits achieve, or almost achieve, this, but bits 5 & 7 and bits 11 & 13 do not. For a full description of the key scheduling algorithm, and terminology see [5].

From Table 3.10, we can see that the key bit XORed with bit 7 on iteration i is XORed with bit 5 on iteration $i+1$, for most iterations i. The same is true for bits 11 & 13. This bit feedback phenomenon will occur if the inverse under permutation PC-2 of any ordered pair of the form

$$(5+6n \bmod 48, 7+6n \bmod 48) \text{ or } (6+6n \bmod 48, 8+6n \bmod 48) \quad 0 \leq n \leq 7$$

consists of a pair (α,β) such that the bit β will be left-shifted to bit α, that is, $\beta = \alpha + 2$, where addition is wraparound on each 24-bit half.

For a full permutation on the 48 bits, this could happen for any of the 16 pairs specified above. In fact, this can only happen on 12, rather than 16 of the pairs, because permutation PC-2 is not really a full permutation on the 48 bits, but two permutations on 24 bits, the two halves being kept separate. Therefore, it is impossible for cloned bits which end up in different halves to possess this property. Permutation PC-2 is specified in Table 3.11 by listing the new location of each of the 48 bits. So the bit in position 1 gets sent to position 14, the bit in position 2 gets sent to position 17 etc. It can be seen that the only two instances are PC-2(1,3)=(5,7) and PC-2(21,23)=(11,13).

We now calculate the probability of this feature of PC-2 having arisen by chance. Let M be the number of pairs in a 24 bit permutation with this property. Each pair has has a probability of $\frac{1}{24}$ of having an inverse of the required form, so under the approximate assumption that the pairs map independently, the distribution of M will be a binomial distribution with parameters $6, \frac{1}{24}$ or approximately Poisson with parameter $\frac{1}{4}$. We observe two realisations of M, with values 0 and 2. The standard hypothesis test that these are drawn from a Poisson($\frac{1}{4}$) would assess 0+2 against the Poisson($\frac{1}{2}$) distribution. A simple calculation yields $\mathcal{P}[X \geq 2] = 0.09$, so the results are not significant occurring with a probability of approximately 9%. If anything, the lack of independence between the pairs will increase this probability.

Therefore, this feature of permutation PC-2 could, and almost certainly did, arise by chance, but it is surprising that the designers of DES did not select a permutation without this property. The result is that bits 5 & 7 and bits 11 & 13 are only XORed with a total of 18 bits of key, rather than the maximum 32. Of course, the permutation of the outputs of the S-boxes stops this effect from

being local, and bits 5 & 7 and 11 & 13 still depend on all the key bits when the cipher is considered as a whole.

Class representative	Size of orbit under P_3
0	1
1	1
x_1	6
x_1x_2	12
$x_1x_2x_3$	8
$x_1 + x_2$	6
$x_1 + x_2x_3$	24
$x_1 + x_2 + x_1x_2$	12
$x_1 + x_2 + x_3$	2
$x_1x_3 + x_2x_3$	12
$x_1 + x_1x_2 + x_2x_3 + x_1x_3$	24
$x_1 + x_2x_3 + x_1x_2x_3$	24
$x_1x_3 + x_2x_3 + x_1x_2x_3$	24
$x_1x_2 + x_2x_3 + x_1x_3$	8
$x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3$	8
$x_1 + x_2x_3 + x_1x_2$	24
$x_1 + x_1x_2 + x_1x_3 + x_2x_3$	4
$x_1 + x_2 + x_1x_2x_3$	24
$x_1 + x_2 + x_3 + x_1x_3 + x_1x_2x_3$	8
$x_1 +_2 + x_3 + x_2x_3 + x_1x_3$	12
$x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3$	4
$x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3$	8

Table 3.1: Equivalence classes when $n = 3$

Size of Class	Frequency	Total
1	2	2
2	1	2
8	6	48
12	3	36
16	16	256
24	5	120
32	11	352
48	27	1296
64	40	2560
96	66	6336
192	166	31872
384	59	22656
		65536

Table 3.2: Distribution of equivalency class sizes when $n = 4$

n	$P[f$ and g lie in same orbit$]$
1	$\frac{3}{8}$
2	$\frac{27}{128}$
3	$\frac{2231}{32768}$
4	$\frac{7836347}{2147483648}$

Table 3.3: The probabilities that two polynomials lie in the same orbit

	Columns															
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Table 3.4: S-box 1

S-box	Row	Bit	Function	S-box	Row	Bit
1	0	2	1110010000111001	5	1	3
2	0	2	1001100101011010	7	0	2
2	1	1	0100101110010110	3	1	3
2	1	4	1101100000100111	3	1	4
3	2	4	1001011011001001	5	3	3
4	0	2	1110010000010111	4	1	1
4	0	3	1011010101001001	4	1	4
4	2	2	0100101110011001	4	3	1
4	2	3	1100011010110100	4	3	4
6	3	4	0100111010110001	8	0	3
7	0	3	0111100010010110	8	1	1
7	0	1	0101101101100100	7	2	2
7	2	3	0010011111100001	8	3	1

Table 3.5: Locations and descriptions of identical twins

S-box	Row	Bit	Function	S-box	Row	Bit
2	0	1	1011010010011001	6	1	4
2	1	2	0111100110001001	2	3	2
2	2	1	0101101001101001	7	1	1
4	0	1	0110001100101101	4	1	2
4	0	4	1101001010011001	4	1	3
4	2	1	1010110110010010	4	3	2
4	2	4	0010011111101000	4	3	3
7	0	4	0100100110111001	7	2	4
8	0	2	1001110000011011	8	1	2

Table 3.6: Locations and descriptions of fraternal twins

S-box	Row	Bit	S-box	Row	Bit	S-box	Row	Bit	Function
2	1	4	3	1	4	3	0	4	1101100000100111
7	0	3	8	1	1	8	0	4	0111100010010110
7	2	3	8	3	1	8	2	4	1101100000011110

Table 3.7: Locations and descriptions of triplets

	x_1	x_2	x_3	x_4
Incidence of a bias of 0	90	69	89	85
Incidence of a bias of 2	38	59	39	43

Table 3.8: Incidences of biases with respect to input bits

	0	1	2	3	4	Total
Predicted frequency	1.9	14.2	39.7	49.2	22.9	127.9
Observed frequency	2	5	50	56	15	128

Table 3.9: Frequency of 0,1,2,3, and 4 biases of 0 per S-box function

Iteration	5	7	11	13
1	49	33	19	3
2	41	25	11	60
3	25	9	60	44
4	9	58	44	57
5	58	42	57	41
6	42	26	41	25
7	26	10	25	9
8	10	59	9	58
9	2	51	1	50
10	51	35	50	34
11	35	19	34	18
12	19	3	18	2
13	3	52	2	51
14	52	36	51	35
15	36	49	35	19
16	57	41	27	11

Table 3.10: Key bits XORed with cloned bits 5 & 7 and 11 & 13

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Table 3.11: Permutation PC-2

Chapter 4

Iterated Functions in Modular Arithmetic

4.1 Introduction

The iteration of simple functions can rapidly produce chaotic behaviour in the fields of real and complex numbers. In this chapter we examine polynomial iteration in the integers modulo N and examine the resulting structures. For each point c in the complex plane the map $f_c : C \to C$ is defined by $f_c(z) = z^2 + c$. The Mandelbrot Set is defined to be those points $c \in C$ for which the sequence $c, f_c(c), f_c(f_c(c)), \ldots$ does not diverge. See [1] for further details.

By considering similar sequences in the integers modulo N, and amending the concept of divergence, we obtain finite analogues of the Mandelbrot Set when N

is the product of certain primes. We are immediately drawn to the application of results concerning elliptic curves defined over finite fields. We only consider iteration by quadratic functions. Iteration by higher degree polynomial functions remains an open area, the arithmetic of higher genus curves being more complex.

4.2 Definitions

Let $\mathbf{Z}_N$ be the ring of integers modulo N. For fixed $N > 0$, define $f_c : \mathbf{Z}_N \rightarrow \mathbf{Z}_N$ by

$$f_c(x) = x^2 + c \bmod N \qquad \text{for } 0 \leq c < N. \tag{4.1}$$

Define s_c to be the infinite sequence $< f_c(c), f_c(f_c(c)), \ldots, f_c^n(c), \ldots >$, where f_c^n denotes n applications of f_c. After at most N iterations, the iterated map must repeat a previously encountered value, and the sequence will enter a recurring loop. We define k to be the smallest k such that $f_c^k(c) = f_c^l(c) \quad k > l \geq 1$. Then we can write

$$s_c =< f_c(c), f_c^2(c), \ldots, f_c^{l-1} > \frown < f_c^l(c), \ldots, f_c^{k-1}(c) >^{\infty}, \tag{4.2}$$

where $\frown$ denotes sequence concatenation, and $< \ldots >^{\infty}$ denotes the sequence repeated *ad infinitum*. If $l = 1$ then we can write $s_c =< f_c(c), \ldots, f_c^{k-1}(c) >^{\infty}$ and say that s_c is cyclic, with fundamental period $(k-1)$. If s_c is cyclic for all $0 \leq c < N$ then we say that N is perfectly cyclic.

4.3 Preliminary Results

Theorem 4.1

If N is perfectly cyclic and M divides N, then M is perfectly cyclic.

Proof

For all c, $0 \leq c < N$, there exists k_c such that $f_c^{k_c}(c) = f_c(c) \bmod N$. Therefore, $f_c^{k_c}(c) = f_c(c) \bmod M$, since M divides N. $f_c(c)$ must be the first repeated value, for if some other value was first repeated, then the loop could not contain $f_c(c) \bmod M$ – a contradiction.

Theorem 4.2

2, 3, and 5 are perfectly cyclic.

Proof

Follows immediately from direct calculation.

4.4 Quadratic Residues and Non–Residues

For a full treatment see [11]. We write $\mathbf{Z}_p^*$ for the group of units of the integers modulo p, where p is prime. An element $x \in \mathbf{Z}_p^*$ is a quadratic residue if $x = y^2 \bmod p$ for some $y \in \mathbf{Z}_p^*$, otherwise x is a quadratic non–residue. The quadratic residues form a normal subgroup of index 2 in the multiplicative group $\mathbf{Z}_p^*$, and so we have the following rules of multiplication:

- Two quadratic residues multiplied together give a quadratic residue.
- Two quadratic non–residues multiplied together give a quadratic residue.
- A quadratic residue and a non–residue multiplied together give a quadratic non–residue.

The Legendre symbol of $x \bmod p$, written $(x \mid p)$, is defined for p odd as

$$(x \mid p) = \begin{cases} 1 & \text{if } x \text{ is a quadratic residue} \\ -1 & \text{if } x \text{ is a quadratic non–residue} \end{cases}$$

for p prime, and $x \in \mathbf{Z}_p^*$.

For p odd, precisely half of the numbers in $\mathbf{Z}_p^*$ are quadratic residues, namely $1^2 \bmod p, 2^2 \bmod p, \ldots, ((p-1)/2)^2 \bmod p$.

4.5 Determination of Perfectly Cyclic Numbers

Theorem 4.3

Let p be an odd prime. If there exists $c (0 < c < p)$ such that $-c$ and $-2c$ are both quadratic non-residues mod p then s_c is not cyclic, and hence p is not perfectly cyclic.

Proof

Consider the repeating loop in s_c. Each value, v, in the loop must have a predecessor, $pre(v)$, such that $(pre(v)^2 + c) \bmod p = v$. For s_c to be cyclic, $f_c(c) = (c^2 + c) \bmod p$ must be present. There are only two possibilities for

$pre(c^2 + c)$, namely $+c$ and $-c$. If $pre(c^2 + c) = +c$, then $pre(+c) = 0$ and $pre(0) = \pm\sqrt{-c}$. If $pre(c^2 + c) = -c$, then $pre(-c) = \pm\sqrt{-2c}$. So if $-c$ and $-2c$ are both quadratic non-residues, the loop cannot contain $f_c(c)$. Hence, s_c could not be cyclic.

Theorem 4.4

Let p be an odd prime. If 2 is a quadratic residue mod p, then p is not perfectly cyclic.

Proof

Consider the $(p-1)$ ordered pairs $(1,2),(2,4),\ldots,(p-1,2p-2)$. Since precisely half the numbers in $\mathbf{Z}_p^*$ are quadratic residues, the average number of quadratic residues per pair is precisely 1. Now take any quadratic residue mod p, x say, and consider the pair $(x,2x)$. Since x and 2 are both quadratic residues, so is $2x$, and the pair contains 2 quadratic residues. Therefore there must exist at least one pair $(y,2y)$ such that y and $2y$ are both quadratic non-residues, otherwise the average number of quadratic residues per pair could not be 1. Take $c = -y$ and apply Theorem 4.3.

We can determine for which odd primes 2 is a quadratic residue by applying the well known result (see, for example, [11]),

$$(2\,|\,p) = (-1)^{(p^2-1)/8}$$

Let $p = 8q + r$, where $r \in \{1,3,5,7\}$, then $(2\,|\,p) = 1$ if and only if, $r = 1$ or $r = 7$.

So we have proved that no primes of the form $8q + 1, 8q + 7$ can be perfectly

cyclic. Now we wish to prove that no primes of the form $8q+3, 8q+5$ can be perfectly cyclic – except, of course, for 3 and 5.

Theorem 4.5

Let p be an odd prime such that 2 is a quadratic non–residue mod p. If there exists a $y \in \mathbf{Z}_p^*$ such that $y^2 - y$ and $y^2 + y$ are both quadratic non-residues then p is not perfectly cyclic.

Proof

Assume, for contradiction, that the hypotheses are satisfied but that s_c is perfectly cyclic. Let $c = -y^2 \bmod p$. From Theorem 4.3, we know that at least one of $-c$ and $-2c$ must be a quadratic residue for s_c to be cyclic. As 2 is a quadratic non-residue, only one of them can be a quadratic residue, and it is clearly $-c$. Therefore, $pre(0) = \pm\sqrt{-c} = \pm y$. Notice that $pre(+y) = \pm\sqrt{+y-c} = \pm\sqrt{y^2+y}$, and $pre(-y) = \pm\sqrt{-y-c} = \pm\sqrt{y^2-y}$. We know that $y^2 - y$ and $y^2 + y$ are both quadratic non-residues, therefore the loop cannot continue back any further and so s_c cannot be cyclic. We have a contradiction so the proof is complete.

Theorem 4.6

Let p be a prime greater than 5 such that 2 is a quadratic non–residue mod p. The prime p is not perfectly cyclic.

Proof

It will suffice to prove there exists a y such that $y^2 - y$ and $y^2 + y$ are both quadratic non-residues mod$p$, and then apply Theorem 4.5. Suppose, for a contradiction, that no such y exists. That is, for every $y \in \mathbf{Z}_p^*$ at least one of $y^2 - y$ and $y^2 + y$ is a quadratic residue.

y	$y^2 - y$	$y^2 + y$
2	2	6
3	6	12
4	12	20
5	20	30
9	72	90

Recall that 2 is a quadratic non–residue, therefore, from $y = 2$, it follows that 6 is a quadratic residue. Now $12 = 2 \times 6$ so 12 is a quadratic non–residue. Therefore, from the case $y = 4$, we see that 20 is a quadratic residue. However, $20 = 2^2 \times 5$, so 5 is a quadratic residue. Now $72 = 6^2 \times 2$, so 72 is a quadratic non–residue. Therefore, from the case $y = 9$, we deduce that 90 is a quadratic residue. Notice that $90 = 3^2 \times 10$, so 10 is a quadratic residue, but $10 = 2 \times 5$, therefore 5 is a quadratic non–residue. Thus we have deduced the contradiction that 5 is both a quadratic reside and non–residue, therefore the assumption that no such y exists must be false. Apply Theorem 4.5 to complete the proof. Note that this proof has assumed that $2, 6, 12, 20, 30, 72, 90 \in \mathbf{Z}_p^*$, and so does not apply to the primes $p = 3$ and $p = 5$.

Combining Theorems 4.2, 4.4 and 4.6 we obtain

Theorem 4.7 The numbers 2, 3 and 5 are the only perfectly cyclic primes.

We know that only numbers which have prime factors 2, 3, and 5 can be perfectly

cyclic, from Theorem 4.1. If we can find a power of 2, 3, or 5, which is not perfectly cyclic, then no perfectly cyclic number can have this power, or any higher power as a factor. By direct calculation we find that 8 is perfectly cyclic, but 16 is not; 9 is perfectly cyclic, but 27 is not; and 25 is perfectly cyclic, but 125 is not. These results prove:

Theorem 4.8

Only numbers of the form $2^a \times 3^b \times 5^c$ where $0 \leq a \leq 3, 0 \leq b \leq 2, 0 \leq c \leq 2$ can be perfectly cyclic.

We can verify that $2^3 \times 3^2 \times 5^2 = 1800$ is, in fact, perfectly cyclic and therefore, by Theorem 4.1, all its divisors are. Theorem 4.13, below, also shows that 1800 is perfectly cyclic. Thus we have a complete determination of perfectly cyclic numbers.

Theorem 4.9

The set of perfectly cyclic numbers is precisely

$$\{2^a \times 3^b \times 5^c \text{ such that } 0 \leq a \leq 3, 0 \leq b, c \leq 2\}$$

$$= \{1, 2, 3, 4, 6, 8, 9, 10, 12, 15, 18, 20, 24, 25, 30, 36, 40, 45, 50,$$

$$60, 72, 75, 90, 100, 120, 150, 180, 200, 225, 300, 360, 450, 600, 900, 1800\}.$$

4.6 General Quadratic Iteration Functions

Having determined the perfectly cyclic numbers for the maps $x \rightarrow x^2 + c$, we would like to generalise to quadratic maps of the form $x \rightarrow ax^2 + bx + c$, where a, b are integers, $a \neq 0$.

First we formally define $f_c : \mathbf{Z}_N \rightarrow \mathbf{Z}_N$ by

$$f_c(x) = ax^2 + bx + c \bmod N \qquad \text{for } 0 \leq c < N. \tag{4.3}$$

As before consider the sequence $s_c =< f_c(c), f_c(f_c(c)), \ldots, f_c^n(c), \ldots >$. If s_c is always cyclic for $0 \leq c < N$ then let us that N is 'perfectly cyclic with respect to $ax^2 + bx$ '. So the original perfectly cyclic numbers become perfectly cyclic with respect to x^2 in this new terminology.

When $a = 0 \bmod p$, the map $f_c(x) = ax^2 + bx + c \bmod p$ becomes linear, and therefore always invertible, so for a given $ax^2 + bx$ all primes which divide a are automatically perfectly cyclic with respect to $ax^2 + bx$. We intend to show that apart from 2, 3, 5, 7 (under certain conditions), and primes which divide a, there are no perfectly cyclic primes with respect to $ax^2 + bx$.

Let $f_c : x \rightarrow ax^2+bx+c \bmod M$, $0 \leq c < M$, $g_c : x \rightarrow ax^2+bx+c \bmod N$, $0 \leq c < N$, and $h_c : x \rightarrow ax^2 + bx + c \bmod MN$, $0 \leq c < MN$.

First we prove

Theorem 4.10

$h_c^i(c) \bmod M = f_{c \bmod M}^i(c \bmod M)$ for $i > 0$.

Proof

Let $c' = c \bmod M$, and $c'' = c \bmod N$. $h_c(y) \bmod M = (ay^2 + by + c \bmod MN) \bmod M = (a(y \bmod M)^2 + b(y \bmod M) + c) \bmod M = f_{c'}(y \bmod M)$ for all $0 \leq y < MN$. Now we can prove the Theorem by induction on i.

For our base case take ($i = 1$). We certainly have that $h_c(c) \bmod M = f_{c'}(c')$ from above taking $c = y$. Now suppose that the the proposition is true for some $i = I$, $I > 0$, then

$$h_c^I(c) \bmod M = f_{c'}^I(c'),$$

$$h_c^{I+1}(c) \bmod M = h_c(h_c^I(c)) \bmod M = f_{c'}(h_c^I(c) \bmod M)$$

$$= f_{c'}(f_{c'}^I(c')) = f_{c'}^{I+1}(c').$$

Hence, by induction, we are done.

Similarly,

$$h_c^i(c) \bmod N = g_{c''}^i(c'') \quad \text{for all } i \geq 1.$$

Theorem 4.11

If M and N are coprime, perfectly cyclic numbers with respect to $ax^2 + bx$, then MN is perfectly cyclic with respect to $ax^2 + bx$.

Proof

With f, g, h, c', c'' defined as above, pick an arbitrary $c, 0 \leq c < MN$, and let k and l be the smallest $k, l > 1$ such that $f_{c'}^k(c') = c'$ and $g_{c''}^l(c'') = c''$, and consider

$$h_c^{\lambda(k-1)(l-1)+1}(c) \text{ where } \lambda \text{ is an integer.}$$

$$h_c^{\lambda(k-1)(l-1)+1}(c) \bmod M = f_{c'}^{\lambda(k-1)(l-1)+1}(c') = f_{c'}(c')$$

$$h_c^{\lambda(k-1)(l-1)+1}(c) \bmod N = g_{c''}^{\lambda(k-1)(l-1)+1}(c'') = g_{c''}(c'')$$

but $h_c(c) \bmod M = f_{c'}(c')$ and $h_c(c) \bmod N = g_{c''}(c'')$, so by the Chinese Remainder Theorem,

$$h_c(c)^{\lambda(k-1)(l-1)+1} = h_c(c),$$

since the two equations have a unique solution modulo MN. So the sequence produced by iteration returns to $h_c(c)$ at least every $(k-1)(l-1)$ values, therefore MN is perfectly cyclic. In fact the fundamental period will be the lowest common multiple of $(k-1)$ and $(l-1)$. This trivially generalises to

Theorem 4.12

If $N_1, \ldots, N_t$ are pairwise relatively prime and are all perfectly cyclic numbers with respect to ax^2+bx, then $\prod_1^t N_i$ is perfectly cyclic with respect to ax^2+bx.

Theorem 4.13

If $p_1, \ldots, p_t$ are the only perfectly cyclic primes with respect to ax^2+bx, and $r_1, \ldots, r_t$ are such that $p_i^{r_i}$ is perfectly cyclic, but $p_i^{r_i+1}$ is not, for each i, then the perfectly cyclic numbers with respect to ax^2+bx are precisely the factors of $\prod_1^t p_i^{r_i}$.

Proof

Apply Theorem 4.12 to the perfectly cyclic primes, with Theorem 4.7 guaranteeing that no other numbers could be perfectly cyclic.

4.7 The Distribution of Quadratic Residues in $\mathbf{Z}_p^*$

For a given prime p, and $0 < d < p$, define

$$Q_d = \{x^2 \mid x^2 + d = y^2 \bmod p, x, y \in \mathbf{Z}_p^*\}.$$

Theorem 4.14

Let r and s be, respectively, a square and a non-square of $\mathbf{Z}_p^*$, then $\#Q_r + \#Q_s = \frac{p-3}{2}$, where $\#$ denotes the size of a set.

Proof

It is well known that $\mathbf{Z}_p^*$ is cyclic (see, for example, [11]). Let g be a primitive root modulo p, that is, a generator of $\mathbf{Z}_p^*$. The squares of $\mathbf{Z}_p^*$ are $\{g^{2k}\}$ and the non-squares are $\{g^{2k+1}\}$.

$$x^2 + r = y^2 \Longleftrightarrow x^2g^2 + rg^2 = y^2g^2 \Longleftrightarrow x^2g^4 + rg^4 = y^2g^4 \Longleftrightarrow \ldots$$

so there is a one-to-one correspondence between the elements of Q_r and Q_{rg^2}, and Q_{rg^4} etc. Therefore, Q_r has the same size for all squares r. Similarly, Q_s has the same size for all non-squares s, but

$$\sum_{d \in \mathbf{Z}_p^*} \#Q_d = \frac{(p-1)(p-3)}{4}$$

since each of the $\frac{p-1}{2}$ squares appears in precisely $\frac{p-3}{2}$ Q-sets.

$$\sum_{d \in \mathbf{Z}_p^*} \#Q_d = \sum_{r \text{ square}} \#Q_r + \sum_{s \text{ non-square}} \#Q_s$$

$$= \frac{p-1}{2}\#Q_r + \frac{p-1}{2}\#Q_s = \frac{(p-1)(p-3)}{4}$$

Therefore, $\#Q_r + \#Q_s = \frac{p-3}{2}$.

Theorem 4.15

If p is of the form $4k+3$, then $\#Q_d = \frac{p-3}{4}$ for all d. If p is of the form $4k+1$, then $\#Q_d = \frac{p-5}{4}$ if d is a square, and $\#Q_d = \frac{p-1}{4}$ if d is a non-square.

Proof

Let $x^2 \in Q_d$, then $x^2 + d = y^2$ for some y. Let $y = x + l$, then

$$x = \left[\frac{d-l^2}{2l}\right] \text{ and } x^2 = \left[\frac{d-l^2}{2l}\right]^2.$$

As l ranges from 1 to $p-1$, x^2 will always be an element of Q_d unless $x^2 = 0$ or $x^2 = -d$, since both x^2 and y^2 must be non-zero. Now we must determine when two different values for the parameter l give rise to the same value of x^2. We have

$$\left[\frac{d-l^2}{2l}\right]^2 = \left[\frac{d-m^2}{2m}\right]^2 \text{ which means that } \frac{d-l^2}{2l} = \pm\frac{d-m^2}{2m}.$$

Therefore, $l = m$ or $lm = -d$ or $l = -m$ or $lm = d$. So, in general, four different values for l of the form $l, -l, d/l, -d/l$ give rise to the same element of Q_d, so we would expect $\#Q_d$ to be approximately $\frac{p-1}{4}$. These four values are distinct unless $l^2 = \pm d$, when they coalesce to just two values. However,

$$l^2 = \pm d \text{ forces } x^2 = \frac{[d \mp d]^2}{\pm 4d} = 0, -d,$$

so the cases $l^2 = \pm d$ do not give valid solutions for x^2.

If $p = 4k + 3$, then $(-1 \,|\, p) = (-1)^{2k+1} = -1$. Therefore, (-1) is a non-square modulo p, and precisely one of d and $-d$ will be a square, so precisely two values of l do not give a valid solution. The remaining $p - 3$ values split into groups of four giving the same value for x^2, so $\#Q_d = \frac{p-3}{4}$.

If $p = 4k + 1$, then (-1) is a square modulo p, and so either $l^2 = d$ and $l^2 = -d$ are both possible, or neither is possible. If d is a square, then both are, so four values for l will be eliminated giving $\#Q_d = \frac{p-5}{4}$. If d is a non-square, then $\#Q_d = \frac{p-1}{4}$.

Corollary

Symmetrically we can define N_d as the set of all non-squares x for which $x + d$ is also a non-square. By precisely the same argument as above we can deduce that if $p = 4k + 3$ then $\#N_d = \frac{p-3}{4}$, and if $p = 4k + 1$ then $\#N_d = \frac{p-5}{4}$ if d is a non-square, and $\#N_d = \frac{p-1}{4}$ if d is a square. Therefore, for all p and d, $\#Q_d + \#N_d = \frac{p-3}{2}$.

4.8 Calculation of Square Distribution using Elliptic Curves

Now we wish to consider $\#(Q_d \cap Q_e)$.

Theorem 4.16

The average value of $\#(Q_d \cap Q_e)$, as d and e range over all non-equal values in

$\mathbf{Z}_p^*$ is

$$\frac{(p-3)(p-5)}{8(p-2)}.$$

Proof

Each of the $\frac{p-1}{2}$ squares has $\frac{p-3}{2}$ neighbours and therefore appears in $\binom{\frac{p-3}{2}}{2}$ different instances of $Q_d \cap Q_e$. Therefore,

$$\sum_{d,e \in \mathbf{Z}_p^*, d \neq e} \#(Q_d \cap Q_e) = \binom{\frac{p-3}{2}}{2} \frac{p-1}{2}$$

$$= \frac{(p-1)(p-3)(p-5)}{16}. \tag{4.4}$$

There are $\binom{\frac{p-1}{2}}{2}$ possibilities for the pair d and e, so

$$\text{Average value of } \#(Q_d \cap Q_e) = \frac{(p-1)(p-3)(p-5)}{16\binom{\frac{p-1}{2}}{2}} = \frac{(p-3)(p-5)}{8(p-2)}. \tag{4.5}$$

So the average value of $\#(Q_d \cap Q_e)$ is approximately $\frac{p}{8}$. We will now establish upper and lower bounds for $\#(Q_d \cap Q_e)$, showing that in fact it is close to $\frac{p}{8}$ for all values of d, e with $d \neq e$.

Theorem 4.17

Provided that $d \neq e$,

$$\frac{p-15}{8} - \frac{\sqrt{p}}{2} \leq \#(Q_d \cap Q_e), \#(N_d \cap N_e) \leq \frac{p}{8} + \frac{\sqrt{p}}{2}.$$

Proof

First we introduce some new notation. Let $S, N, \star$ stand for a square in $\mathbf{Z}_p^*$, a non-square in $\mathbf{Z}_p^*$ and any element of $\mathbf{Z}_p^*$ respectively. For given d and e with

$d \neq e, de \neq 0$ we can use the alphabet $\{S, N, \star\}$ to write a three letter codeword representing the number of triples $(x, x+d, x+e)$ where the first, second and third symbols state the restrictions placed on $x, x+d$ and $x+e$ respectively. For example, we write $SN\star$ for the number of triples $(x, x+d, x+e)$ where x is a square in $\mathbf{Z}_p^*$, $x+d$ a non-square in $\mathbf{Z}_p^*$, and $x+e$ any element of $\mathbf{Z}_p^*$.

From the definitions of the symbols we immediately obtain the following rewriting rules for expressions involving $\star$

$$SS\star = SSS + SSN \tag{4.6}$$

$$S\star S = SSS + SNS \tag{4.7}$$

$$\star SS = SSS + NSS \tag{4.8}$$

$$NN\star = NNS + NNN \tag{4.9}$$

$$N\star N = NSN + NNN \tag{4.10}$$

$$\star NN = SNN + NNN. \tag{4.11}$$

In this new notation $\#(Q_d \cap Q_e)$ becomes SSS and $\#(N_d \cap N_e)$ becomes NNN. Recall that $SS\star$ is the number of triples $(x, x+d, x+e)$ where x and $x+d$ must be squares but $x+e$ can be any element of $\mathbf{Z}_p^*$. This will be equal to $\#Q_d$ unless $(-e)$ and $(-e+d)$ are both squares when $(-e) \in Q_d$ but $(-e, -e+d, 0)$ is not a valid triple since $0 \notin \mathbf{Z}_p^*$. This could be avoided by amending the definition of $\star$ to be any element of $\mathbf{Z}_p$, but then the rewriting rules involving $\star$ would no longer hold.

Therefore,

$$SS\star = \begin{cases} \#Q_d - 1 & \text{if } (-e) \in Q_d \\ \#Q_d & \text{otherwise.} \end{cases} \tag{4.12}$$

Similarly,

$$NN\star = \begin{cases} \#N_d - 1 & \text{if } (-e) \in N_d \\ \#N_d & \text{otherwise.} \end{cases} \tag{4.13}$$

Combining (4.12) and (4.13) and noting that $(-e) \notin Q_d \cap Q_e$, we deduce that

$$\#Q_d + \#N_d - 1 \leq SS\star + NN\star \leq \#Q_d + \#N_d. \tag{4.14}$$

From the Corollary to Theorem 4.15 we know that $\#Q_d + \#N_d = \frac{p-3}{2}$, giving us

$$\frac{p-3}{2} - 1 \leq SS\star + NN\star \leq \frac{p-3}{2}. \tag{4.15}$$

In the same way we can show that $S\star S + N\star N$ and $\star SS + \star NN$ lie in the same range.

Now we can use an elliptic curve argument to estimate SSS and NNN. For a full treatment of elliptic curves see [4] and [12]. To find SSS we must count the number of $x \in \mathbf{Z}_p^*$ for which $x, x+d, x+e$ are all non-zero squares. Consider the elliptic curve $y^2 = x(x+d)(x+e)$ over $\mathbf{Z}_p$. By Hasse's theorem, the number, U, of points on the curve satisfies $|U - (p+1)| \leq 2\sqrt{p}$. Let U' be the number of distinct non-zero x-co-ordinates of points on the curve, then $U' = \frac{U-4}{2}$, since we are not counting the "point at infinity", or the three points with y-co-ordinate 0, and the remaining points are of the form $(x, \pm y)$. Using the possible range for U given by Hasse's Theorem we deduce that

$$\frac{p-3}{2} - \sqrt{p} \leq U' \leq \frac{p-3}{2} + \sqrt{p}. \tag{4.16}$$

U' counts those points for which $x, x+d, x+e$ are all squares and those for which precisely one of the three terms is a square. In our new notation this means that

$$U' = SSS + SNN + NSN + NNS. \tag{4.17}$$

Now we use (4.9)–(4.11) and (4.17) to rewrite (4.16) as

$$\frac{p-3}{2} - \sqrt{p} \leq SSS + (\star NN + N \star N + NN\star) - 3NNN \leq \frac{p-3}{2} + \sqrt{p}. \quad (4.18)$$

U' is the number of values of $x \in \mathbf{Z}_p - \{0, -d, -e\}$ for which $x(x+d)(x+e)$ is a square. It is clear that the number of values of $x \in \mathbf{Z}_p - \{0, -d, -e\}$ for which $x(x+d)(x+e)$ is a non-square, U'' say, also lies in the range $\frac{p-3}{2} \pm \sqrt{p}$ since obviously $U' + U'' = p - 3$. Thus we obtain the symmetrical counterpart of (4.18):

$$\frac{p-3}{2} - \sqrt{p} \leq NNN + (\star SS + S \star S + SS\star) - 3SSS \leq \frac{p-3}{2} + \sqrt{p}. \quad (4.19)$$

Solving (4.18) and (4.19) simultaneously and using (4.12)-(4.14) to evaluate the terms involving $\star$, we deduce that

$$\frac{p-15}{8} - \frac{\sqrt{p}}{2} \leq SSS, NNN \leq \frac{p}{8} + \frac{\sqrt{p}}{2}. \quad (4.20)$$

Substituting (4.12),(4.13), (4.18) and (4.19) into (4.6)–(4.11) we obtain a complete determination of all our three letter codes:

$$\frac{p-18}{8} - \frac{\sqrt{p}}{2} \leq SSN, SNS, NSS, NNS, NSN, SNN \leq \frac{P+10}{8} + \frac{\sqrt{p}}{2}. \quad (4.21)$$

We can now use this result about the distribution of the squares and non-squares in $\mathbf{Z}_p^*$ to prove that perfectly cyclic primes with respect to $ax^2 + bx$ cannot exist for certain values of a and b.

4.9 Perfectly Cyclic Primes

Theorem 4.18

For a given a and b, if c can be found such that $b^2-4ac, b^2-4ac-4b, b^2-8ac-4b$ are all non-squares modulo prime p, then p is not perfectly cyclic with respect to ax^2+bx.

Proof

As in the proof of Theorem 4.3, consider the backwards chain of predecessors of ac^2+bc+c: $pre(ac^2+bc+c) = c, -\frac{b}{a}-c$ and then $pre(c) = 0, -\frac{b}{a}$ and $pre(-\frac{b}{a}-c) = \frac{-b\pm\sqrt{b^2-8ac-4b}}{2a}$. Finally $pre(0) = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$ and $pre(-\frac{b}{a}) = \frac{-b\pm\sqrt{b^2-4ac-4b}}{2a}$. If there exists an instance of c, such that b^2-4ac, $b^2-4ac-4b$, and $b^2-8ac-4b$ are all non-squares, then the chain of predecessors cannot be followed back to ac^2+bc+c, and so p is not perfectly cyclic with respect to ax^2+bx.

We will now show that for sufficiently large p, it is always possible to find such a c, unless $b = \pm 4 \bmod p$ or $b = 0 \bmod p$. Let $\Delta = b^2-4ac$. We must find $\Delta, \Delta-4b, 2\Delta-4b-b^2$ all non-squares. There are values of c for which these three quantities are not all different namely $c=0$ and $c=-\frac{b}{a}$. However, the case $c=0$ would force Δ to be a square, and $c=-\frac{b}{a}$ forces $\Delta-4b$ to be a square, so neither special value of c could be included when counting up instances of $\Delta, \Delta-4b, 2\Delta-4b-b^2$ all being non-squares. Consider the elliptic curve over $\mathbf{Z}_p^*$ defined by

$$y^2 = 2x(x-4b)(x-\frac{(b^2+4b)}{2}). \tag{4.22}$$

Let V be the number of values of $\Delta \in \mathbf{Z}_p^*$ for which $\Delta, \Delta-4b, 2\Delta-4b-4b^2$ are all non-squares. The presence of an extra factor of 2 makes a slight difference. If 2 is a square then $V = NNN$ whereas if 2 is a non-square then $V = NNS$. From (4.20) and (4.21) we see that there is no real difference between NNN

and NNS, only slightly more uncertainty in the value of NNS. Taking $d = -4b, e = -\frac{(b^2+4b)}{2}$ and applying Theorem 4.17,

$$\frac{p-18}{8} - \frac{\sqrt{p}}{2} \leq V \leq \frac{p+10}{8} + \frac{\sqrt{p}}{2}. \tag{4.23}$$

This assumes that $d, e \neq 0 \bmod p$ and that $d \neq e \bmod p$. The only cases for which these conditions do not hold are $b = 0 \bmod p$ and $b = \pm 4 \bmod p$.

Now V is the number of suitable values of Δ for which $\Delta, \Delta - 4b$ and $2\Delta - 4b - b^2$ are all non-squares, so when $\frac{p-18}{8} > \frac{\sqrt{p}}{2}$, we have $V > 0$, and we can always choose c so that Δ is one of the values counted by V. The condition that $\frac{p-18}{8} > \frac{\sqrt{p}}{2}$ simplifies to $p > (2+\sqrt{22})^2 \approx 45.7$.

There are still a couple of loose ends to be tied up. The proof of Theorem 4.18 does not apply to the cases $b = 0, \pm 4 \bmod p$. This corresponds to the fact that if 2 is a non-square in $\mathbf{Z}_p^*$ then it is not possible to ever find $-4ac, -4ac, -8ac$ all non-squares when $b = 0$ since precisely one of $-4ac$ and $-8ac$ must be a square, and similarly we can never find $16 - 4ac, -4ac, -8ac$ all non-squares when $b = 4$, or $16 - 4ac, -4ac, 32 - 8ac$ all non-squares when $b = -4$. We are as yet unable to deal with the case $b = \pm 4$ satisfactorily.

4.9.1 The Special Case $b = 0 \bmod p$

We can deal with the case $b = 0 \bmod p$. The initial value for each cycle is $ac^2 + c$. As in the proof of Theorem 4.3, $pre(ac^2 + c) = \pm c$, $pre(+c) = 0$, $pre(-c) = \pm\sqrt{-\frac{2c}{a}}$ and $pre(0) = \pm\sqrt{-\frac{c}{a}}$.

When 2 is a square mod p then we choose c so that $-c$ and $-2c$ are both squares if a is a non-square, or both non-squares if a is a square. The proof of Theorem

4.4 guarantees that we can always find such a c. Now both $-\frac{c}{a}$ and $-\frac{2c}{a}$ are non-squares and so we cannot continue taking predecessors back any further. Therefore we can never return to $ac^2 + c$ and no prime p for which 2 is a square mod p can be perfectly cyclic with respect to ax^2.

When 2 is a non-square mod p we can always find a y such that $y^2 + y$ and $y^2 - y$ are both non-squares as in the proof of Theorem 4.6. Let $c = -\frac{y^2}{a}$. Only one of $-\frac{c}{a}$ and $-\frac{2c}{a}$ can be a square and in this case it is clearly $-\frac{c}{a} = \frac{y^2}{a^2}$. Now $pre(\pm\sqrt{-\frac{c}{a}}) = \pm\sqrt{\frac{\pm\sqrt{-\frac{c}{a}}-c}{a}} = \pm\sqrt{\frac{\pm\frac{y}{a}+\frac{y^2}{a}}{a}} = \pm\frac{\sqrt{\pm y+y^2}}{a}$. We already know that $y^2 + y$ and $y^2 - y$ are both non-squares, therefore we cannot continue taking predecessors back any further. Therefore we can never return to $ac^2 + c$ and no prime p for which 2 is a non-square can be perfectly cyclic with respect to $ac^2 + c$, except possibly 3 and 5. [Recall that the proof of Theorem 4.6 applies to all primes p for which 2 is a non-square mod p, except for $p = 3$ and $p = 5$.] By direct calculation it is easily verified that 2, 3 and 5 are perfectly cyclic with respect to ax^2.

We have proved that when $b = 0 \bmod p$ the only perfectly cyclic primes with respect to $ax^2 + bx$ are 2, 3, 5, and any prime which divides a.

4.9.2 The Special Case $b = \pm 4 \bmod p$

As observed above, the proof technique of Theorem 4.14 does not work for the case $b = \pm 4 \bmod p$ as we cannot ever find three non-squares in the required form if 2 is a non-square modulo p. By following the iteration back to the next predecessor we can derive the following expressions which would all have to be non-squares or non-existent for the case $b = 4 \bmod p$: $2 - ac \pm \sqrt{4 - ac}$,

$2 - ac \pm \sqrt{-2ac}$, and $2 - ac \pm \sqrt{-ac}$. Similarly, for the case $b = -4 \bmod p$ we would need to find $6 - ac \pm \sqrt{4 - ac}$, $6 - ac \pm \sqrt{8 - ac}$, and $6 - ac \pm \sqrt{8 - 2ac}$ all non-square or non-existent.

Theorem 4.19

If we can find $y \in \mathbf{Z}_p^*$ such that $2 + y^2 + y$, $2 + y^2 - y$, and $4 + y^2$ are all non-squares, then p cannot be perfectly cyclic with respect to $ax^2 + b$ when $b = \pm 4 \bmod p$.

Proof

The problems only occur when 2 is a non-square modulo p so we will assume that this is the case, otherwise the Theorem is true trivially.

For the case $b = 4 \bmod p$ we choose c such that $-ac = y^2$. Now $\sqrt{-2ac}$ is non-existent, and the other two terms become $2 + y^2 \pm \sqrt{4 + y^2}$ and $2 + y^2 \pm y$. From the conditions on y we deduce that all three terms are non-existent or non-squares.

Similarly for the case $b = -4 \bmod p$ we choose c such that $4 - ac = y^2$. So $\sqrt{8 - 2ac}$ is non-existent, and other two terms become $2 + y^2 \pm y$ and $2 + y^2 \pm \sqrt{4 + y^2}$ precisely as before. Once again we deduce that all three terms are non-existent or non-squares.

So in either case we cannot continue the iteration back any further, and therefore p cannot be perfectly cyclic with respect to $ax^2 + bx$.

How likely is it that we can find such a y? As y varies over $\mathbf{Z}_p^*$ each of the terms $2 + y^2 + y$, $2 + y^2 - y$ and $4 + y^2$ will be a non-square precisely half the time.

If the terms were independent of each other then we would expect them to all be non-squares $\frac{p-1}{8}$ of the time. The number of suitable y has been counted for each prime with 2 a non-square for all $p < 1000$. The results for $p < 320$ are shown in the graph below against the line $y = \frac{p-1}{8}$, with similar behaviour occurring when $p > 320$. As the graph shows, there are indeed close to $\frac{p-1}{8}$ suitable values of y.

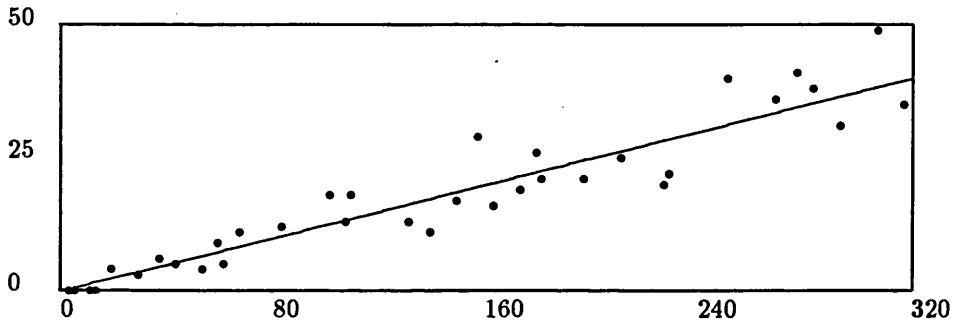

Figure 4.1: The number of suitable y for each prime of the correct form

Recall that we only need to exhibit one value of y so that the three terms are all non-squares, and in general it appears we have about $\frac{p-1}{8}$. We make the following conjecture:

Conjecture 4.1

For prime $p > 43$ of the form $8k+3$ or $8k+5$ we can always find $y \in \mathbf{Z}_p^*$ so that $2+y^2+y$, $2+y^2-y$, and $4+y^2$ are all non-squares, and therefore deal with the case $b = \pm 4 \bmod p$.

Summarising the above results we have

Theorem 4.20

For prime $p \geq 47$, and non-zero $b \neq \pm 4 \bmod p$, we can always choose c so that $\Delta, \Delta - 4b, 2\Delta - 4b - b^2$ are all non-squares, and therefore by Theorem 4.18, p is not perfectly cyclic with respect to $ax^2 + bx$. Furthermore, when $b = 0$, then p is only perfectly cyclic with respect to ax^2 if $p = 2, 3, 5$ or p divides a.

When $b = \pm 4 \bmod p$ then p is very probably only perfectly cyclic with respect to $ax^2 + bx$ if $p = 2, 3, 5$ or p divides a. This is not a theorem, but a conjecture.

Finally we deal with the remaining primes less than 47. For a fixed prime, p, we can determine the values of a and b, for which p will be perfectly cyclic. Simply compute the sequences $< f_c(c), f_c^2(c), \ldots, >$ for each combination of values for $a \bmod p$ and $b \bmod p$, and test for perfect cyclicity. This means we can test the primes up to 43 individually, then invoke Theorem 4.5 to dispose of the rest. Calculation shows that 2, 3, 5 are always perfectly cyclic; 7 is perfectly cyclic if and only if, $b = \pm 2 \bmod 7$; and no other prime $p < 47$ is perfectly cyclic with respect to any $ax^2 + bx$ unless $a \bmod p = 0$.

4.10 Perfectly Cyclic Prime Powers

Having determined the perfectly cyclic primes for a general quadratic map we can now consider the problem of determining which powers of each perfectly cyclic prime are also perfectly cyclic. There are 2 possibilities; either there exists a k such that $p, p^2, \ldots, p^k$ are perfectly cyclic and p^{k+1} is not, or p^k is perfectly cyclic for all k. If we can find a p^k which is not perfectly cyclic then

all higher powers are automatically not perfectly cyclic. The general question of determining the highest perfectly cyclic power of a perfectly cyclic prime is unsolved, and the power seems to depend on a and b in a chaotic way. However we may investigate two special forms of quadratic, the cases $b = 0$ and $a = 1$.

Table 4.1 summarises the highest perfectly cyclic prime powers for each perfectly cyclic prime with respect to ax^2 for small values of a. Table 4.2 does likewise with respect to $x^2 + bx$ for small values of b. A prime to the power ∞ means that all powers of the prime are perfectly prime.

From the tables we can conclude that powers of 2, 3 and 5 occurring as factors in a tend to increase the highest perfectly cyclic power of these primes, but the behaviour is complex, even for small a and b. We can prove one result suggested by Table 4.2.

Theorem 4.21

For odd b, all powers of 2 are perfectly cyclic with respect to $x^2 + bx$.

Proof

We will prove the theorem by induction. We prove that if 2^N is perfectly cyclic with respect to $x^2 + bx$ for some N, then so is 2^{N+1}.

Define $f(x) = x^2 + bx \bmod 2^N$ and $f'(x) = x^2 + bx \bmod 2^{N+1}$. If 2^N is perfectly cyclic with respect to $x^2 + bx$ then for all $c : 0 \leq c < 2^N$ the sequence $f_c(c), f_c(c)^2, f_c(c)^3, \ldots$ is cyclic. That is, there exists a k such that $f_c(c)^k = f_c(c)$. Now consider $f'_c(c)^k$. No earlier value in the loop could have been equal to $f'_c(c)$ since no two earlier terms are equal mod2^N. Ei-

ther $f'_c(c)^k = f'_c(c)$ when we have a cycle, or $f'_c(c)^k = (f'_c(c) + 2^N) \bmod 2^{N+1}$. Now we have that $f'_c(c)^{k+1} = (f_c(c)^2 + b2^N) \bmod 2^{N+1}$. Therefore, when b is odd, $f'_c(c)^{k+1} = (f_c(c)^2 + 2^n) \bmod 2^{N+1}$. Repeating this argument we deduce that $f'_c(c)^{k+2} = (f_c(c)^3 + 2^N) \bmod 2^{N+1}$, $f'_c(c)^{k+3} = (f_c(c)^4 + 2^N) \bmod 2^{N+1}$, $\ldots, f'_c(c)^{2k-1} = (f_c(c)^k + 2^N) \bmod 2^{N+1} = f'_c(c)$. So in either case 2^{N+1} is also perfectly cyclic. We know that 2 is always perfectly cyclic, hence by induction, all powers of 2 are perfectly cyclic with respect to $x^2 + bx$ when b is odd.

4.11 Conclusions on Perfectly Cyclic Primes

For a given polynomial $ax^2 + bx$ (where $b \neq \pm 4 \bmod p$) the primes which are perfectly cyclic with respect to $ax^2 + bx$ are: 2, 3, 5, 7 if and only if, $b = \pm 2 \bmod 7$, and any primes which divide a.

The same primes are also perfectly cyclic when $p = \pm 4 \bmod p$, but there remains a small possibility of more primes also being perfectly cyclic with respect to $ax^2 + bx$.

a	Highest perfectly cyclic power
1	$2^3, 3^2, 5^2$
2	$2^4, 3^2, 5^2$
3	$2^3, 3^2, 5^2$
4	$2^5, 3^2, 5^2$
5	$2^3, 3^2, 5^2$
6	$2^4, 3^2, 5^2$
7	$2^3, 3^2, 5^2, 7^2$
8	$2^7, 3^2, 5^2$
9	$2^3, 3^4, 5^2$
10	$2^4, 3^2, 5^2$
11	$2^3, 3^2, 5^2, 11^2$
12	$2^5, 3^2, 5^2$
13	$2^3, 3^2, 5^2, 13^2$
14	$2^4, 3^2, 5^2, 7^2$
15	$2^3, 3^2, 5^2$
16	$2^9, 3^2, 5^2$
17	$2^3, 3^2, 5^2, 17^2$
18	$2^4, 3^4, 5^2$
19	$2^3, 3^2, 5^2, 19^2$
20	$2^5, 3^2, 5^2$

Table 4.1: Highest powers of perfectly cyclic primes when $b = 0$

b	Highest perfectly cyclic power
0	$2^3, 5^2, 3^2$
1	$2^\infty, 3^2, 5^2$
2	$2^4, 3^3, 5, 7$
3	$2^\infty, 3^2, 5$
4	$2^3, 3^2, 5$
5	$2^\infty, 3^2, 5, 7$
6	$2^4, 3^2, 5^2$
7	$2^\infty, 3^3, 5$
8	$2^3, 3^2, 5$
9	$2^\infty, 3^2, 5^2$
10	$2^4, 3^2, 5^2$
11	$2^\infty, 3^4, 5^2$
12	$2^3, 3^2, 5, 7$
13	$2^\infty, 3^2, 5$
14	$2^4, 3^2, 5^2$
15	$2^\infty, 3^2, 5^2$
16	$2^3, 3^4, 5^2$
17	$2^\infty, 3^2, 5$
18	$2^4, 3^2, 5$
19	$2^\infty, 3^2, 5^2$
20	$2^3, 3^3, 5^2$

Table 4.2: Highest powers of perfectly cyclic primes when $a = 1$

Acknowledgements

I would like to thank my supervisor Geoff Smith for much advice and encouragement throughout. I would also like to thank James Davenport and Icarus Sparry for their expert help on many occasions.

Bibliography

[1] B. Branner, "The Mandelbrot Set", in Robert L. Devaney and Linda Keen (eds.), *Chaos and Fractals: The mathematics behind the computer graphics*, volume 39 Proceedings of Symposia in Applied Mathematics, AMS, 1989.

[2] E. F. Brickell, J. H. Moore, and M. R. Purtill, "Structure in the S-boxes of the DES" in *Advances in Cryptology, Proceedings of Crypto 86, Lecture Notes in Computer Science*, Springer Verlag, 1987, pp 3-8.

[3] J. Cannon, *A Language for Group Theory*, Department of Pure Mathematics, University of Sydney, 1982.

[4] J. W. S. Cassels, *"Lectures on Elliptic Curves,"* CUP, 1991.

[5] Dorothy E. Denning, *Cryptography and Data Security*, Addison-Wesley, 1982.

[6] Y. Desmedt, J-J. Quisqauter, and M. Davio, "Dependence of output on input in DES: Small avalanche characteristics",*Advances in Cryptology, Proceedings of Crypto 84, Lecture Notes in Computer Science*, Springer Verlag, 1985, pp 359-376.

[7] R. Forré, "The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition" in *Advances in Cryptology, Proceedings of Crypto 88, Lecture Notes in Computer Science*, Springer Verlag, 1988, pp 450-468.

[8] M. Hall, *The Theory of Groups*, Macmillan, 1959.

[9] A. Hearn, *REDUCE User's Manual, Version 3.4*, RAND Corporation Publication CP-78, 1991.

[10] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard", *Information Systems Laboratory, Stanford University*, 1976.

[11] E. Kranakis, *Primality and Cryptography*, Wiley-Teubner, 1986

[12] S. Lang, *Elliptic Curves and Diophantine Analysis*, Springer-Verlag, 1978.

[13] S. A. Lloyd, "Balance, uncorrelatedness and the Strict Avalanche Criterion", *Hewlett-Packard Research Laboratories, Bristol Technical Memo HPL-ISC-TM-89-012*, 1989.

[14] S. A. Lloyd, "Characterising and counting functions satisfying the Strict Avalanche Criterion of order (n-3)", *Proceedings of the Second IMA Conference on Cryptography and Coding*, 1989.

[15] C. Mitchell, "Enumerating boolean functions of cryptographic significance," *Journal of Cryptology*, 2(3): pp 155-170, International Association for Cryptologic Research, 1990.

[16] A. O. Morris, *Linear Algebra*, Van Nostrand Reinhold 1978.

[17] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govarts and J. Vandewalle, "Propagation Characteristics of Boolean Functions" in *Advances in Cryptology, Proceedings of Eurocrypt 90, Lecture Notes in Computer Science*, Springer Verlag, 1991, pp 161-173.

[18] R. Rueppel, "Correlation immunity and the summation generator" in *Advances in Cryptology, Proceedings of Crypto 85, Lecture Notes in Computer Science*, Springer Verlag, 1986, pp 260-272.

[19] A. Shamir, "On the security of DES" in *Advances in Cryptology, Proceedings of Crypto 85, Lecture Notes in Computer Science*, Springer Verlag, 1986, pp 280-281.

[20] T. Siegenthaler, "Correlation Immunity of Non-Linear Combining Functions for Cryptographic Applications", *IEEE Transactions on Information Theory*, volume IT-30, 1984, pp 776-779.

[21] T. Siegenthaler, "Correlation-immune polynomials over finite fields" in *Advances in Cryptology, Proceedings of Eurocrypt 86, Lecture Notes in Computer Science*, Springer Verlag, 1987.

[22] A. F. Webster and S. E. Tavares, "On the design of S-boxes" in *Advances in Cryptology, Proceedings of Crypto 85, Lecture Notes in Computer Science*, Springer Verlag, 1988, pp 523-534.

[23] H. Wielandt, *Finite Permutation Groups*, Academic Press, New York, 1964.

Appendix A

Cayley code

A.1 Code for the classification of polynomials under the group P_n

The following code sets up the vector space of polynomials in 2 variables over the field Z_2, and was used for the 3 and 4 variable case with obvious amendments.

```
"First we set up our polynomials as a vector space
over the field k and generate the group."
k=field(2);
v=vector space(4,k);
g=general linear(v);
a=mat(
1,0,0,0:
```

```
0,0,1,0:
0,1,0,0:
0,0,0,1) of g;
b=mat(
1,0,0,0:
1,1,0,0:
0,0,1,0:
0,0,1,1) of g;
c=mat(
1,0,0,0:
0,1,0,0:
1,0,1,0:
0,1,0,1) of g;
h=<a,b,c>; print h; print order(h);

"Now we initialise our vectors by simply looping
over all members of the space"
vecset=[vec(0,0,0,0) of v];
for b1=0 to 1 do
for b2=0 to 1 do
for b3=0 to 1 do
for b4=0 to 1 do
vecset=vecset join [vec(b1,b2,b3,b4) of v];
end; end; end; end;

"float contains to set of unclassified vectors,
```

```
for each element of the set we compute its orbit
and remove the orbit from float"
float=vecset;
for each x in vecset do
if x in float then do
temp=[x];
for each y in h do
temp=temp join [x*y];
float=float-[x*y];
end;
print temp; print card(temp);
end; else loop; end;
end;
```

A.2 The procedures implementing the equivalence testing algorithm

The following are all the CAYLEY procedures used by the equivalence testing algorithm.

```
n = numofvbles;

"Input non-negative integer q; output binary rep of q in reverse order
 as a sequence"
procedure binrep(q;bin);
```

```
bin = empty;
while q ne 0 do
r = q mod 2;
q = (q-r)/2;
bin = append(bin,r);
end;
end;

"A pre-computation. In effect this sets up a sequence containing
all possible monomials in the n variables, each stored as a binary sequence"
refstore = empty;
for i = 0 to 2^n-1 do
binrep(i;tem);
refstore = append(refstore, tem);
end;

" take a monomial and replace vbl by vble + 1. Either monomial is
fixed, then output is -1, else add new monomial called poly."
procedure reflectmono(i,mon;poly);
binrep(mon;bin);
poly = -1;
if i le length(bin)  then
if  bin[i] eq 1 then
poly = mon -2^(i-1);
end;
end;
```

```
end;

"apply a reflection to each monomial in a given polynomial;"
procedure reflpoly(i,poly;newpoly);
newpoly = poly;
store = empty;
for each p in poly do
reflectmono(i,p;q);
if q ne -1 then
store = append(store,q);
end;
end;
for j = 1 to length(store) do
if store[j] in newpoly then
newpoly = newpoly - [store[j]];
else newpoly = poly join [store[j]];
end;
end;
end;
"computes a sequence showing the number of monomials of various
degrees, starting with degree 0"
procedure shapetest(poly,n;shape);
shape = conseq(0,n+1);
for each x in poly do
binrep(x;binx);
wt = 0;
```

```
for i = 1 to length(binx) do
wt = wt + binx[i];
end;
shape[wt+1] = shape[wt+1] + 1;
end;
end;

"takes binary sequences, and pads them with zeros to make them
have length n. Padding is on the right."
procedure padout(n,seqofseqs;pad);
pad = empty;
for each x in seqofseqs do
y = x;
while length(y) lt n do
y = append(y,0);
end;
pad = append( pad, y);
end;
end;
"takes a binary sequence in reverse order and turns it into base 10"
procedure bintonum(binseq;mono);
mono = 0;
for i = 1 to length(binseq) do
if binseq[i] eq 1 then
mono = mono + 2^(i-1);
end;
```

```
end;
end;
"takes a sequence of binary sequences, each in reverse order,
and converts each of the constituent sequences into base 10,
and puts these base 10 numbers into a set called poly"
procedure binseqtonum(seqofseqs;poly);
poly = null;
for each x in seqofseqs do
bintonum(x;temp);
poly = poly join [temp];
end;
end;
"searches to see if two similarly shaped polynomials are equivalent
under the action of the symmetric group on the letters"
procedure equivsearch(n,poly1,poly2;flag);
flag = false;
bp1 = empty;
for each mon in poly1 do
binrep(mon;binstring);
bp1 = append(bp1,binstring);
end;
g = symmetric(n);
padout(n,bp1;padbp1);
for each x in g do
per = eltseq(x);
temp = padbp1;
```

```
bpnew = empty;
for each y in padbp1 do
z = y;
for i = 1 to length(z) do
z[per[i]] = y[i];
end;
bpnew = append(bpnew,z);
end;
binseqtonum(bpnew;poly);
if poly eq poly2 then
flag = true;
print 'the permutation ',per, ' does the trick';
break;
end;
end;
if flag eq false then
print 'these polynomials are not equivalent';
end;
end;
" sets up a list of small primes for use in hashing"
m = 0;
prim = empty;
p = 1;
while length(prim) le n+1 do
p = p+1;
if not prime(p) then
```

```
loop;
end;
prim = append(prim,p);
end;

"input a shape of a polynomial -- a sequences telling you
the number of monomials of given weight in ascending order,
and a list of primes prim -- output a single number, a hash
of the shape"
procedure hash(shap,prim;has);
has = 1;
for i = 1 to length(shap) do
has = has*prim[i]^shap[i];
end;
end;

poly = [2^n-1 , 2^(n-1)];

"Input the number of variables n, a polynomial, a list
of all possible monomials refstore, and small primes prim;
output is hashbucket"
procedure hashes(n,poly,refstore,prim;hashbucket);
hashbucket = null;
for i = 1 to length(refstore) do
tempoly = poly;
for j = 1 to length(refstore[i]) do
```

```
if refstore[i][j] eq 1 then
reflpoly(j,tempoly;tempoly);
end;
end;
shapetest(tempoly,n;form);
hash(form,prim;has);
hashbucket = hashbucket join [has];
end;
```